سلسلة كن أسدا للإبداع

BE LION CREATIVITY

سبيلك المختصر إلى تعلم

لغة C#.NET

برمجة الواجهات

خالد السعداني

سبيلك المختصر لتعلم لغة

C#.Net

من إعداد : خالد السعداني

"يا أيها الذين آمنوا اتقوا الله و قولوا قولا سديدا.

يصلح لكم أعمالكم و يغفر لكم ذنوبكم

ومن يطع الله و رسوله فقد فاز فوزا عظيما"

الأحزاب : 70 و 71

إهداء

إلى الوالدين أولا وأخيرا...

إلى أحبائي متتبعي السلسلة...

إلى كل المسلمين والمسلمات...

اللهم اجعله عملا خالصا لوجهك

عند وجود أي ملاحظة،

المرجو مراسلتي عبر :

Khalid_Essaadani@Hotmail.Fr

الفهرس

BE LION
arabic formation

BE LION
arabic formation

BE LION
arabic formation

تقديم

كنت قد وعدت الإخوة الأفاضل والأخوات الفاضلات الذين اطلعوا على الجزء الأول الخاص بالأساسيات والمفاهيم الرئيسية بجزء ثان يعرض برمجة الواجهات بأسلوب سهل ومستساغ، وقد تأخرت كثيرا لأبدأ في إخراج هذا الجزء بسبب مشاغل الحياة، على العموم هذا هو الكتاب المكمل للجزء الأول الذي آمل أن يكون خير معين لكل مسلم ومسلمة يرغب في تعلم البرمجة بلغة السي شارب.

لعل الإخوة يلاحظون غياب كتب تفصيلية تنطلق بهم من البدء إلى الختم بتدرج يتوافق مع متطلباتهم، طبعا هنالك كُتَّابْ ما شاء الله عليهم نسأل الله أن يزيدهم علما وسدادا، ولكن الخصاص قائم وبارز للعيان، وعليه فإخراج كتاب متواضع كهذا قد يكون لبنة تنضاف إلى صرح البرمجة العربية، نعلم أن أعمالنا يشوبها نقص ويتخللها ضعف ولكن كما يقال "أعمش خير من أعمى"، فكتاب خير من لاشيء، لهذا فأنا أشدد مجددا على الإخوة ليساندونا في تطوير هذه السلسلة ولو بعون بسيط، فقد نادينا منذ سنتين بالمساهمة فيها وللأسف لم أتلق أي رسالة ترمي إلى مشاركتنا في إثراء مكتبة البرمجة العربية.

حتى لا يساء فهم هذا الجزء فأنا أقصد بالمساندة الكتابة والتأليف وليس الجانب المادي، وأرجو من الله العلي القدير أن أستقبل بعد إخراج هذا الكتاب رسالة من أخ أو أخت يريد الإسهام في كتابة

السلسلة حتى وإن كان مبتدئا، فأنا مستعد بعون الله أن أمد إليه يد المساعدة حتى يخرج كتابه في أبهى حلة.

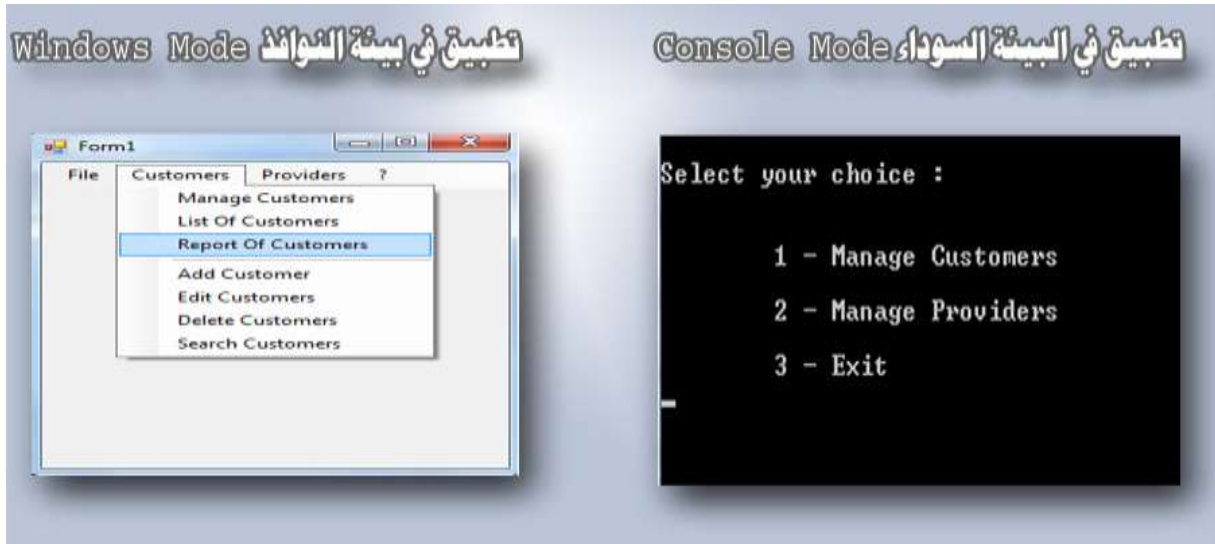
في هذا الكتاب تخليت عن استعمال الكلمات الفرنسية بسبب طلبات الأصدقاء، واستعملت الكلمات الانجليزية نظرا للجمهور العربي العريض الذي يعرف أبجدياتها أكثر من الفرنسية، ستجد صور النوافذ بالفرنسية وهذا بسبب نسخة الفيجوال أستوديو التي أشتغل عليها ولكن لا عليك فالشرح بالإنجليزية يحل المشكلة، وقد حاولت قدر المستطاع تبسيط المفاهيم وتوصيلها بطرق سلسة، فأحيانا ستجدني أسهبت في شرح أشياء عادية ليس حبا في ذلك ولكن ليكون الكتاب عاما يفهمه كل من يقرؤه.

كما يقول الشاعر: لكل شيء إذا ما تم نقصان، فطبعا مجهودنا يبقى مجهودا بشريا، ففي حال عثرتم على خطأ في الكتاب لا تترددوا في مراسلتي به عبر البريد الالكتروني لأقوم بتصحيحه في النسخة القادمة إن شاء الله، إذا احتجتم أي مساعدة فأنا رهن إشارتكم سواء عبر بريدي الالكتروني أو حتى عبر رقم هاتفي في الحالات العاجلة أو لمن لقي مشكلا برمجيا.

أسعد كثيرا برسائلكم، وأسأل الله أن يرزقنا وإياكم الإخلاص في العمل، دام لكم البشر والفرح !

العبد الفقير إلى ربه :خالد السعداني

مدخل إلى برمجة الواجهات

لا غرو أن برامج البيئة السوداء (الدوس) تختلف عن برامج بيئة النوافذ (ويندوز) في طريقة العمل وفي نتائجه، وإن كنا في البيئة السوداء نسعى دوما إلى تسهيل الأمر بالنسبة للمستخدم، إلا أن ذلك يبقى بدائيا بالمقارنة مع هيئة تطبيقات الويندوز، إذ أن للمستخدم إمكانيات عديدة متاحة أمامه من تحريكٍ للماوس وضغطٍ وجذبٍ واختيارٍ وليس كالبيئة السوداء التي تُحَتم عليه الاشتغال بلوحة المفاتيح فقط، وتقييده بتنفيذ سطر قبل المرور إلى غيره، كما سأعرض في المثال التالي ليتضح كلامي أعلاه:

هنا في هذه الصورة نعرض نفس التطبيق وقد أنجزناه في البيئة السوداء وفي بيئة النوافذ، لاحظ أن في البيئة الأولى على المستخدم أن يحدد اختيارا قبل المرور إلى الأمر الموالي، أما في بيئة النوافذ

فللمستخدم كل الحرية في تقديم القائمة التي يشاء وكذلك في التحكم في النافذة بكل حرية إذا أراد إغلاقها أو تحجيمها أو إنزالها إلى شريط المهام.

ملحوظة:

في البيئة السوداء يمكننا تغيير الألوان كما نشاء وإدخال الأصوات أيضا، كما سنرى فيما يلي.

سنقوم بتغيير لون خلفية الخط من الأسود إلى اللون **الأصفر** مثلا، وكذلك لون الخط من الأبيض إلى **الأحمر**.

```
using System;
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.BackgroundColor = ConsoleColor.Yellow;
            Console.ForegroundColor = ConsoleColor.Red;
            Console.Write("Hi Brothers ! ");
            Console.ReadKey();
        }
    }
}
```

النتيجة كما يلي :

```
Hi Brothers !
```

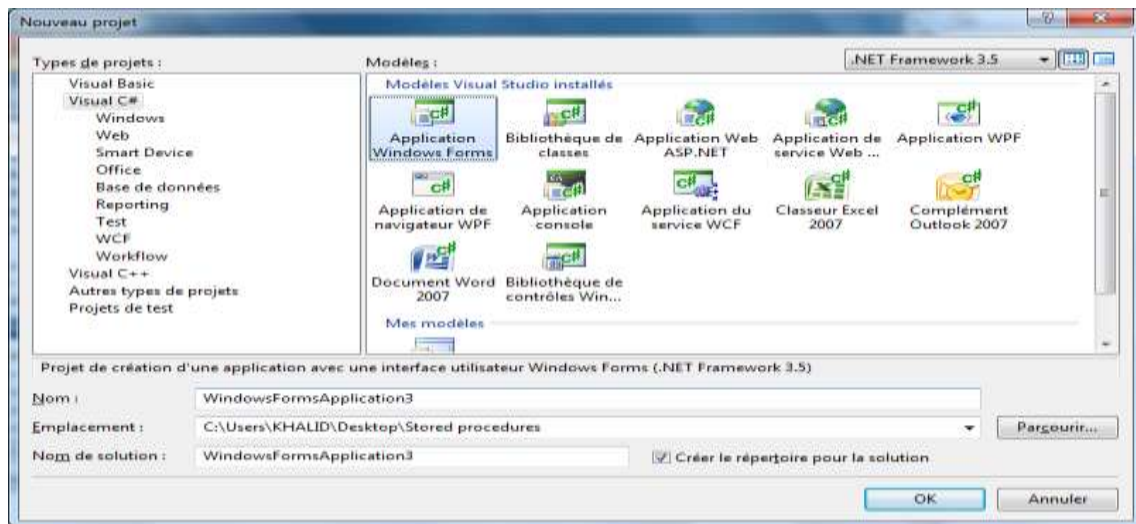
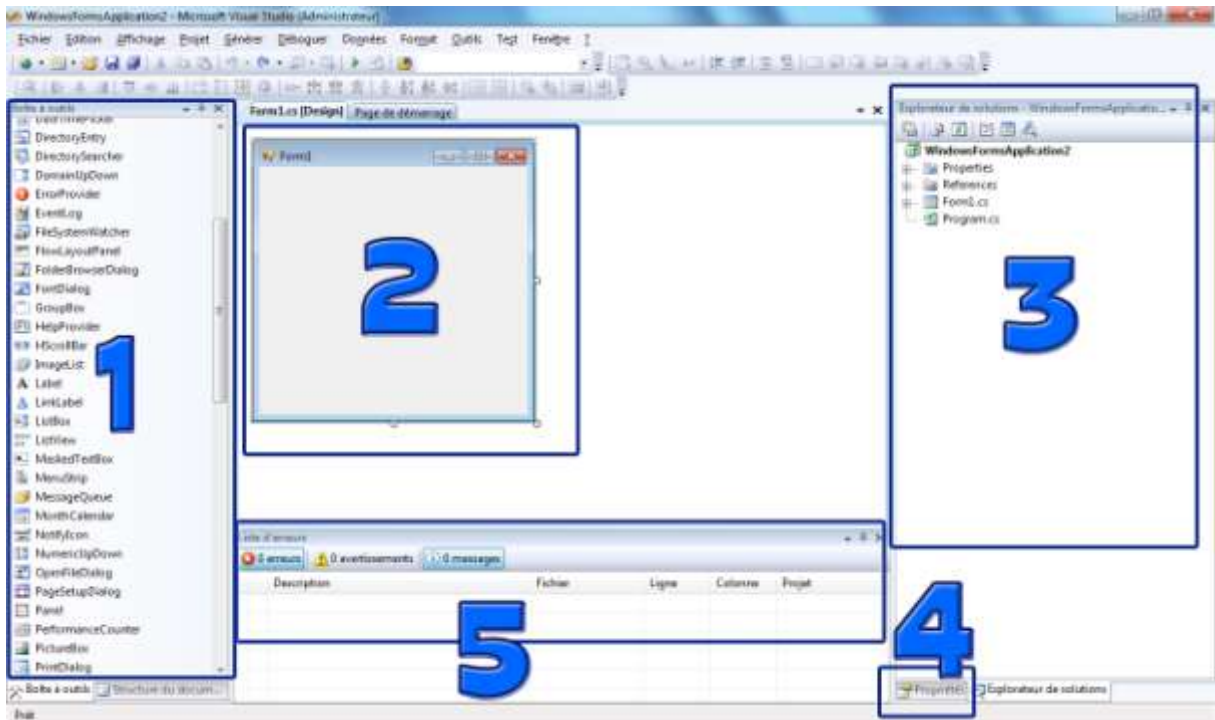
ولإصدار صوت تحذيري :

```
Console.Beep();
```

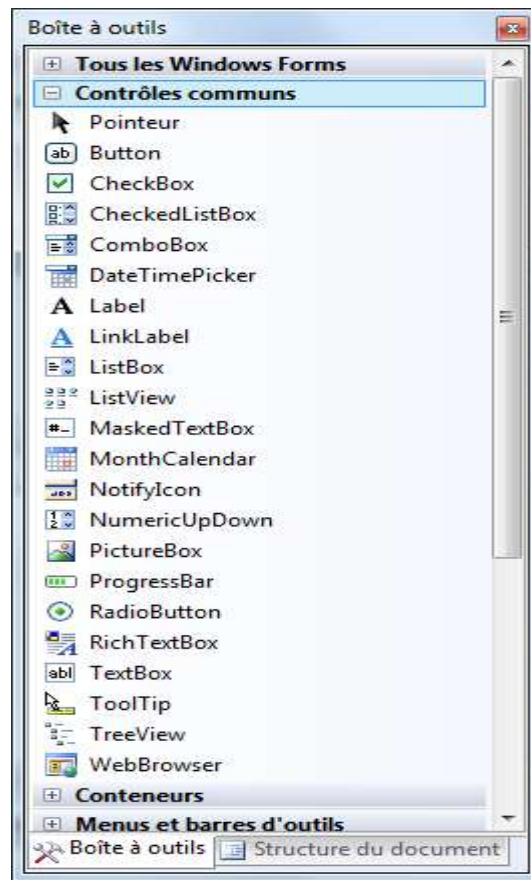
الآن سنغير المحطة، استعد لنبدأ برمجة الواجهات، بسم الله على بركة الله !

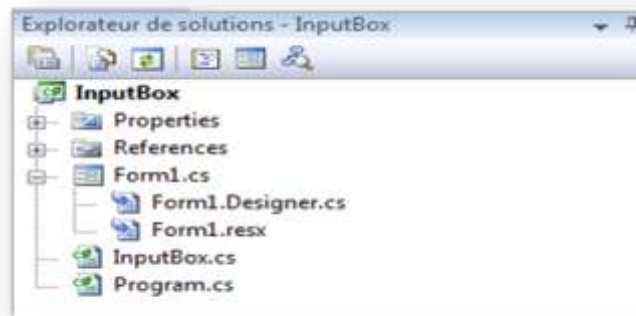
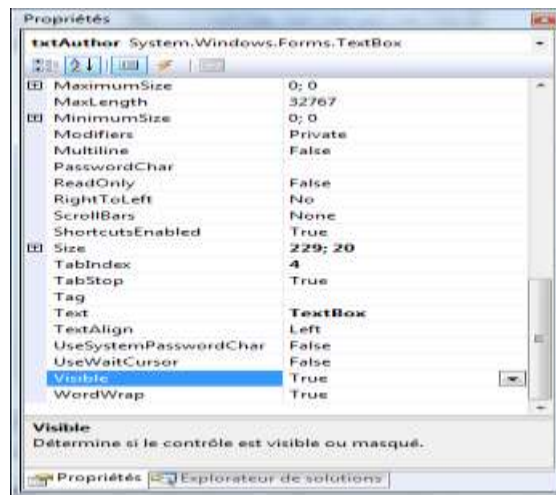
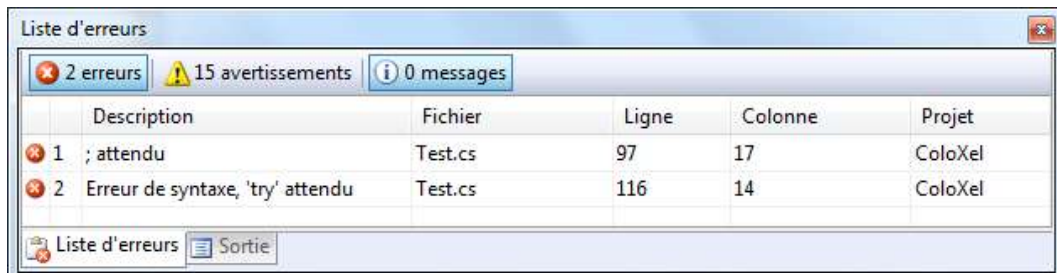
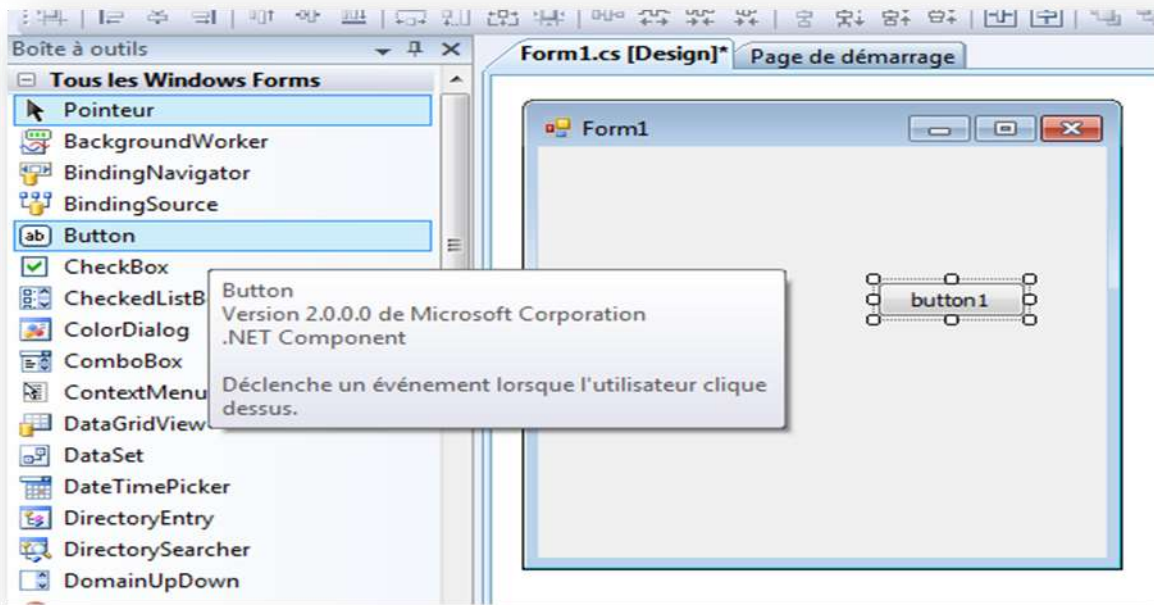
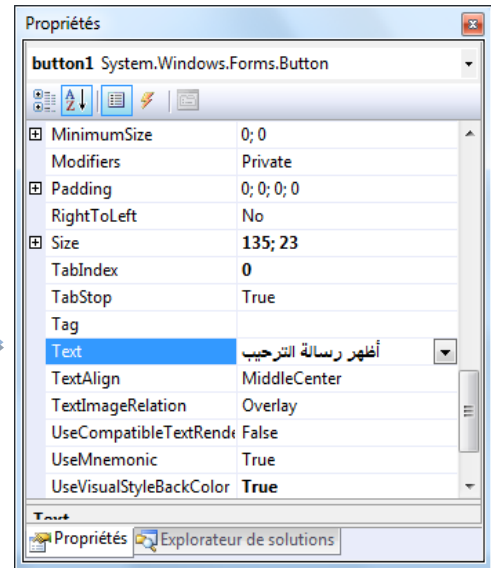
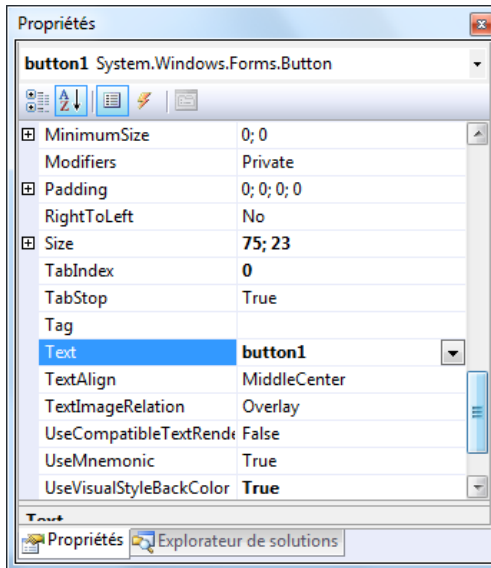
بيئة التصميم Design environment

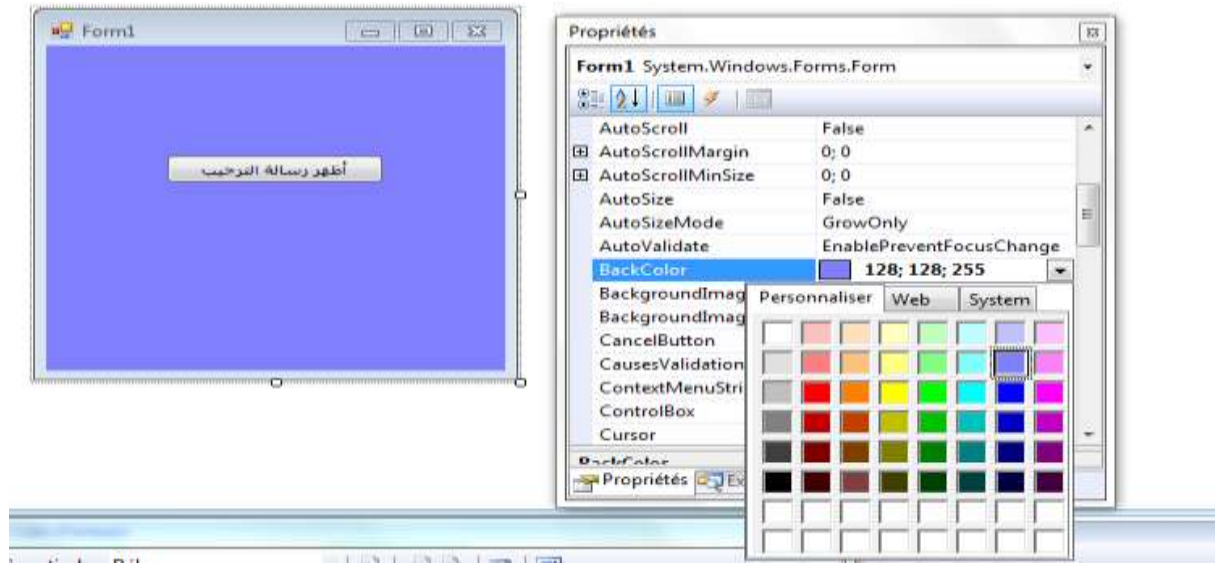
كل البرامج تحتوي على بيئة الاشتغال، وهي بمثابة ورشة تضم كل مكونات البرنامج التي قد يحتاجها المستخدم في عمله.

للولوج إلى نافذة التصميم، سنقوم أولا بإنشاء مشروع جديد وذلك عن طريق فتح برنامج الفيجوال استوديو، ثم الذهاب إلى القائمة File واختيار New project، بعد ذلك ستظهر لنا هذه النافذة، أو بكل بساطة الضغط على Ctrl+Shift+N.

نقوم بتحديد نوع المشروع من اليسار باختيار Visual c# ثم نختار Windows Forms Application لإنشاء مشروع جديد من نوع تطبيق ويندوز، نكتب اسم المشروع ونختار مسار حفظه، ثم نختم بالضغط على OK لتتم إحالتنا مباشرة على بيئة التصميم الخاصة بمشروعنا الجديد.

سنورد شرحا بسيطا لكل جزء من بيئة التصميم حسب الأرقام الواردة في الصورة:

1. وتسمى هذه النافذه بعلبة الأدوات ToolBox، وهي تضم كل الأدوات التي قدم يحتاجها برنامجك (أزرار، قوائم، ...) . إن لم تكن ظاهرة عندك فاذهب إلى القائمة View ثم

اخترها لكي تظهر، أو اكتف بالضغط على الإختصار Ctrl+Alt+X، وهذه صورة لعلبة الأدوات:

2. هذا هو الفورم Form الذي سنضع عليه الأدوات اللازمة لبناء المشروع ويمكنك إضافة العديد من الفورمات إلى مشروعك كما سنرى فيما بعد إن شاء الله، ويمكنك تغيير مقاسه عبر مسك الزوايا وجذبها أو من خلال نافذة الخصائص رقم 4، وهذه صورة لفورم وعليه بعض الأدوات:

3.ويسمى هذا الجزء متصفح المشروع Solution Explorer، وسمي كذلك لأنه يعرض كل الملفات التي يضمها المشروع حسب تبويبات خاصة بكل نوع، ويمكن إظهاره في حالة غيابه عن طريق الذهاب إلى القائمة View واختيار Solutions Explorer أو الإكتفاء بالضغط على الاختصار Ctrl+Alt+L

4. نافذة الخصائص Properties : وتحتوي على خصائص الأداة التي تحددها، ومن خلال هذه النافذة يمكننا تغيير اللون والخلفية و الخط وباقي الخصائص،   لإظهارها في حالة اختفائها قم بتحديد الأداة المرغوب تغيير خصائصها والضغط عليها بيمين الماوس واختيار Properties أو يكفيك الضغط على زر لوحة المفاتيح F4

5. قائمة الأخطاء Error List وتعرض هذه النافذة الأخطاء المرتكبة قبل بدء عملية ترجمة الشفرة Compilation، من خلالها يمكنك معرفة مكان الخطأ ليتأتى لك تصحيحه.

أتمنى أن تكون قد استأنست بمكونات بيئة التصميم وأدركت مهمة كل جزء، الآن تعال بنا ننجز أول تطبيق لنا في عالم برمجة النوافذ، قم بالذهاب إلى علبة الأدوات Toolbox وابحث عن أداة الزر Button ثم اضغط عليها بالماوس واجذبها إلى الفورم بكل سهولة، أو اضغط عليها مرتين لكي تنضاف إلى الفورم.

جيد، الآن قم بتحديد الزر الذي أضفناه للتو، ثم اذهب إلى نافذة الخصائص لنقوم بتعديل النص المكتوب عليه، اسم الخاصية الموكولة بالنص المكتوب على الزر هو Text يمكنك استبدال قيمتها بأي نص تشاء كما يظهر جليا في الصورة التالية:

سيصبح الزر كما يلي :

بنفس الطريقة سنغير لون الفورم، وذلك بتحديده أولا ثم الذهاب إلى الخصيصة BackColor ونختار اللون الذي نشاء.

بإمكانك تغيير بعض قيم الخصائص من باب تجربتها والتعرف عليها، بعد الانتهاء من التصميم سنقوم بكتابة الكود الخاص بإظهار رسالة ترحيب عند الضغط على الزر.

لا ترتبك فالأمر بسيط جدا، يكفيك الضغط مرتين على الزر ليتم نقلك إلى هذه النافذة (نافذة كتابة الأكواد) :

```
WindowsFormsApplication2.Form1    button1_Click(object sender, EventArgs e)
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Data;
 5  using System.Drawing;
 6  using System.Linq;
 7  using System.Text;
 8  using System.Windows.Forms;
 9
10  namespace WindowsFormsApplication2
11  {
12      public partial class Form1 : Form
13      {
14          public Form1()
15          {
16              InitializeComponent();
17          }
18
19          private void button1_Click(object sender, EventArgs e)
20          {
21
22          }
23      }
24  }
```

سنورد شرحا بسيطا لكل سطر مهم في هذه النافذه حتى تتضح الرؤية بالنسبة للإخوان الذين يدخلون عالم البرمجة بلغة السي شارب لأول مرة:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
```

هذه الأسطر تسمى مجالات الأسماء Namespaces وقد شرحناها في الجزء الأول، ولكن لا بأس بإعادة ذلك:

تذكير (مجالات الأسماء):

وهي تلك الأسماء التي تكون مسبوقة بالموجهة using و هي مجالات تضم العديد من الفئات والأنواع، كما يمكن لمجال واحد أن يضم مجموعة من مجالات الأسماء كما هو الحال مع المجال System الذي نجد به فئات كثيرة مثل console و convert وكذلك يضم مجالات أسماء فرعية مثل IO و Collections وأول سطر يكون في البرنامج هو سطر التأشير إلى مجالات الأسماء ويكون باستعمال الأمر الموجه using متبوعا باسم مجال الأسماء ، واستعمال هذه الطريقة يوفر علينا أن نقوم كل مرة بكتابة مجال الأسم قبل فئاته، فلولا هذه الإمكانية لكتبنا System.Console.WriteLine() بدل Console.WriteLine().

ملحوظة 1 :

في الأسطر أعلاه معظم المجالات لن نحتاجها الآن، لذلك يمكنك حذف كل الأسطر ماعدا :

```
using System;
using System.Windows.Forms;
```

ملحوظة 2:

عند الاشتغال على تطبيقات الويندوز لابد من جلب مجال الأسماء System.Windows.Forms، لأنه يضم كل الفئات اللازمة لهذا النوع من المشاريع، ولعل أهم فئة هي System.Windows.Forms.Control لأنها الفئة الأم التي ترث منها معظم الأدوات Controls

مباشرة بعد مجالات الأسماء، يأتي هذا السطر :

```
namespace WindowsFormsApplication2
```

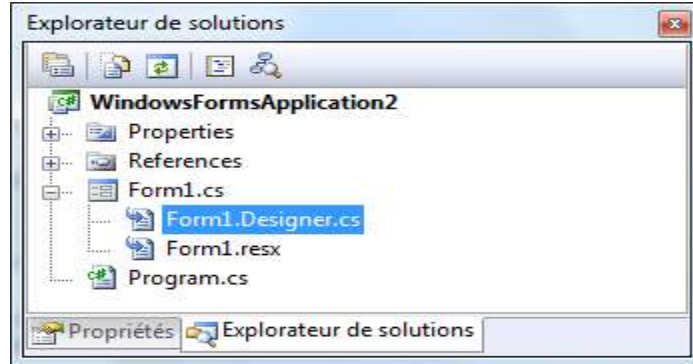
هنا يتم تعريف المشروع كما لو أنه مجال أسماء ليتأتى لنا استغلال فئاته Classes في تطبيق آخر بمجرد عمل using لهذا المجال .

أما هذا السطر :

```
public partial class Form1 : Form
```

فيه نجد مفهومين أحدهما رأيناه في الجزء الأول وهو الوراثة Inheritance، حيث نقوم بوراثة Form1 مشتق من الفئة Form الرئيسية، أما المفهوم الثاني وهو الفئات الجزئية partial classes فإنه يسمح لنا بتقسيم الفئة الواحدة على العديد من الملفات حيث نستعمل الكلمة المحجوزة partial أمام اسم الكلاس إذا احتجنا إلى الاشتغال عليه في ملف آخر.

أرهقتك على ما أعتقد 🙂

طيب اذهب إلى ملف التصميم الخفي Form1.Designer.cs (سميته كذلك لأنه يقوم بترجمة كل ما نقوم به على الفورم إلى شفرة)

انظر إلى ملف التصميم Form1.Designer.cs ، وستشاهد نفس اسم الكلاس Form1

```
namespace WindowsFormsApplication2
{
    partial class Form1
    {
```

انظر إلى ملف الكود وستشاهد نفس اسم الكلاس Form1

```
namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
```

هل اتضحت الرؤية الآن؟

ليس بعد ؟؟

هل تلاحظ أن نفس الفئة موزعة على ملفين متفرقين، إذا كان ذلك كذلك فقد استوعبت الأمر،

ركز فقط في كلامي أعلاه وستفهم بعون الله !

بعد أن تفهم هذا الجزء تعال معي إلى الأسطر الموالية لنفهمها :

```
public Form1()
        {
            InitializeComponent();
        }
```

السطر الأول يمثل الإعلان عن مشيد Constructor الخاص بالفئة Form1، والدالة التي بداخله IntializeComponent() تقوم بعمل Initialize للفورم ومكوناته أي تعيدها لنقطة البداية، وهذه الدالة معرفة على مستوى الفئة Form1.Designer.cs

تنبيه :

إذا لم تكن قد سمعت بمفهوم المشيدات Constructors عد إلى الجزء الأول لتفهمها بشكل جيد.

أما الدالة التالية:

```
private void button1_Click(object sender, EventArgs e)
        {

        }
```

فقد تم إنشاؤها عندما ضغطنا مرتين على الزر ليتم نقلنا إلى نافذة الكود، في هذا المكان أي كود يكتب سيتم تنفيذه عند القيام بضغط الزر Click.

قلنا سنظهر رسالة ترحيبية، فليكن ذلك إذن، لإظهار علبة الرسائل نكتب السطر التالي داخل اللامتين :

```
    private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("أهلا وسهلا بكم في أول تطبيق عملي في بيئة النوافذ !");
        }
```

نقوم بتنفيذ البرنامج لنشاهد النتيجة:

عندما ستضغط على الزر ستظهر هذه الرسالة الترحيبية.

آمل أن تكون قد نجحت في إنتاج أول تطبيق عملي، وفهمت بالأساس ما يحدث لأنه كان يمكننا إنجاز هذا التطبيق في صفحة واحدة وستنجح في ذلك ولكنك لن تفهم شيئا، أطلنا وأسهبنا حتى

تتعرف على كل جزء وتستوعب كل سطر ومع ذلك فلم نتطرق إلى كل شيء، لكن على العموم رأينا أشياء مهمة لننطلق ونحن على يقين بأننا بصدد البرمجة وليس بصدد تعلم الكتابة 😃

الأدوات Controls

الأدوات هي كل ماستحتاجه لبناء برنامجك، من فورم Form وعلب النص Textbox وأزرار Button وقوائم ListBox وغيرها، وقد رأينا أين توجد هذه الأدوات وقلنا بأنها موجودة في علبة الأدوات Toolbox

سنورد للأدوات قدرا كافيا من التفصيل فيما سيأتي- إن شاء الله - لشرح معظمها والدور المنوط بكل أداة.

الخصائص Properties

لنقرب مفهوم الخصائص إلى أذهاننا، سنأخذ على سبيل المثال بيتا، فالبيت هو بمثابة أداة Control وخصائصه عديدة كلون البيت، ومساحته، وعلوه، و عدد أطباقه وما إلى ذلك، نفس الشيء ينطبق على أداوت البرمجة، فكل أداة تتوفر على خصائص تتميز بها عن غيرها، إضافة إلى أنها تشترك مع باقي الأدوات في مجموعة من الخصائص، سنوردها فيما يلي بعض الخصائص التي قد تحتاجها في عملك:

اسم الخصيصة	دورها
Text	لتغيير النص الظاهر على الأداة
BackColor	لتغيير لون خلفية الأداة
BackgroundImage	لتغيير صورة خلفية الفورم Form
BackgroundImageLayout	كيفية ظهور صورة خلفية الفورم، وتأخذ القيم التالية: None: ستظهر الخلفية في أعلى يسار الفورم. Tile: ستظهر الصورة مكررة طولا وعرضا. Stretch: ستظهر الصورة ممدة على كل الفورم. Zoom: ستظهر الصورة في أقصى مقاسها مركزة في وسط الفورم.
ForeColor	لتغيير لون النص المكتوب على الأداة.
Font	لتغيير الخط المكتوب على الأداة.
Name	لتغيير اسم الأداة، وهذا هو أهم خصيصة للأداة لأننا سنتعامل معه

الخاصية	الوصف
	عند كتابة الشفرة، فلهذا يرجى تذكر هذا جيدا وعدم الخلط بين نص الأداة Text وبين اسم الأداة Name فلا وجه للتشابه بينهما.
TabIndex	ترتيب الأداة إذا استعمل المستخدم زر Tab الموجود على لوحة المفاتيح.
Width	تغيير عرض الأداة.
Height	تغيير علو الأداة (طولها عموديا).
Visible	هذه الخصيصة تقبل إما صح True أو خطأ False، وهي تتحكم في ظهور الأداة أثناء تنفيذ البرنامج، بمعنى إذا كانت قيمة هذه الخصيصة False فلن تظهر الأداة عند التنفيذ.
Enabled	لإلغاء اشتغال الأداة أثناء التنفيذ إذا كانت قيمتها False، ولتفعيل الأداة نجعل قيمتها True
Dock	لتثبيت الأداة في إحدى زوايا الفورم.
Anchor	لجعل مقاس الأداة مرتبطا بمقاس الفورم، وهذا مهم حينما يكون

	الفورم قابلا للتصغير والتكبير فعلى الأدوات الموضوعة عليه أن تكون مرتبطة به ليبدو البرنامج أكثر احترافية.
Parent	لتحديد الفورم الأب للفورم الحالي (سنتعرف على هذا المفهوم فيما سيأتي إن شاء الله)
Cursor	لتغيير شكل مؤشر الماوس عند مروره فوق الأداة.
FormBorderStyle	لتحديد شكل إطار الفورم، من هنا يمكن إختيار القيمة التي تجعل الفور غير قابل لتغيير مقاسه وهي FixedSingle، القيمة FixedDialog تؤدي نفس الدور أيضا مع اختلاف شكل الإطار.
Icon	لتغيير أيقونة الفورم، على شرط أن يكون امتداد الأيقونة ico
Opacity	لتغيير شفافية الفورم، القيمة 0 تجعل الفورم مخفيا، والقيمة 1 تظهره بشكل عادي، وكلما غيرت القيمة من 0 إلى 1 تغيرت الشفافية.
MaximizeBox	لإخفاء زر التكبير الموجود على الشريط العلوي للفورم، وأيضا

	لإظهاره.
MinimizeBox	لإخفاء زر إنزال الفورم إلى شريط المهام، وأيضا لإظهاره.
WindowState	لتغيير حجم الفورم ويأخذ القيم التالية: • Normal: القيمة الافتراضية حيث يظهر الفورم في المقاس الذي صممته عليه. • Maximized: لإظهار الفورم بشكل مكبر يشغل شاشة الحاسوب. • Minimized: لإنزال الفورم إلى شريط المهام.
Location	موضع الفورم حسب الإحداثيات X و Y
StartPosition	الموضع الذي ستظهر فيه النافذه عند التنفيذ، ويأخذ القيم التالية: • CenterParent: ليظهر الفورم في وسط الفورم الأب. • CenterScreen : ليظهر الفورم في وسط

	الشاشة. Manual: لجعل الموضع مرتبطا بالخصيصة Location. WindowsDefaultBounds: لجعل المقاس والموضع مرتبطان بنظام التشغيل. WindowsDefaultLocation: لجعل الموضع لوحده مرتبطا بنظام التشغيل، أما المقاس فيبقى كما حددته عند التصميم.

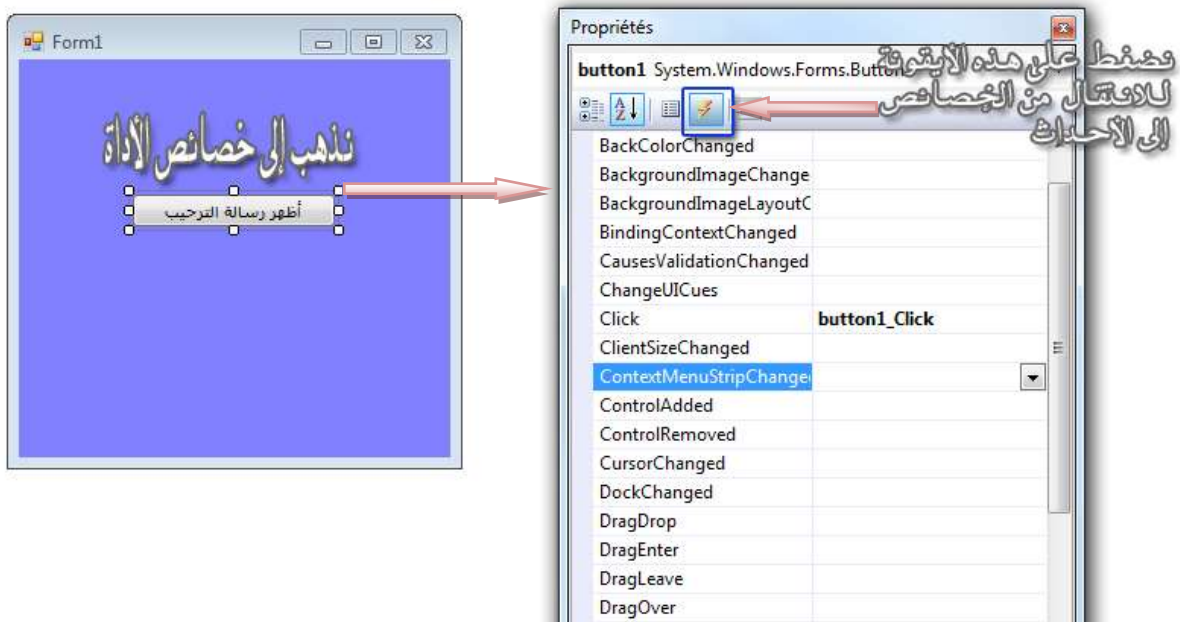
هذه أهم الخصائص التي قد تحتاجها، لم نشأ أن نعرض كل الخصائص تفاديا للإطالة.

الأحداث Events

تعرفنا في الجزء الأول على مفهوم الأحداث في البيئة السوداء، نفس المفهوم أيضا في بيئة النوافذ يعيد نفسه، إلا أنه هذه المرة أكثر تفاعلية ونشاطا ☺

تذكير (الأحداث):

حينما تضغط على زر الماوس، أو على زر من أزرار لوحة المفاتيح مثلا، فإن ذلك يرسل رسالة إلى دالة مرتبطة بهذا الحدث لتقوم بتنفيذها، كما رأينا في المثال العملي الأول، عندما نضغط بالماوس على الزر Button فإن ذلك ينفذ الدالة التي تظهر رسالة ترحيبية.

تتوفر كل أداة على عدد كبير من الأحداث والتي يمكنك رؤيتها في نافذة الخصائص بعد تحديد الأداة المراد مشاهدة أحداثها، كما تظهر الصورة التالية:

كما تلاحظ توجد قائمة طويلة من الأحداث، لكتابة الشفرة التي تريد تنفيذها عند حدث معين يكفيك الضغط على اسم الحدث مرتين ليتم نقلك مباشرة إلى نافذة الكود وبالضبط داخل دالة مرتبطة بهذا الحدث، جرب أي حدث وسترى كيف يحدث ذلك.

سنعرض فيما يلي جدول بعض الأحداث التي قد تحتاجها أثناء إنجازك لبرنامج ما، وأمام كل حدث سنورد وصفه:

الحدث	دوره
Click	ينفذ هذا الحدث حينما نضغط على الأداة بزر الماوس الأيسر مرة واحدة.
DoubleClick	ينفذ هذا الحدث حينما نضغط على الأداة بزر الماوس الأيسر مرتين متتاليتين.
BackColorChanged	يتولد هذا الحدث حينما يقوم المستخدم بتغيير لون خلفية الأداة.
FormClosing	ينفذ هذا الحدث عند إغلاق الفورم.

FormClosed	ينفذ هذا الحدث بعد إغلاق الفورم.
Load	ينفذ هذا الحدث عند بداية تحميل الفورم.
KeyDown	هذا الحدث يتعلق بلوحة المفاتيح، ويتولد حينما نضغط على مفتاح ما، وهو يأتي قبل الحدثين KeyPress و KeyUp
KeyPress	هذا الحدث يتعلق بلوحة المفاتيح أيضا، ويتولد حينما نضغط على مفتاح ما وهو ينفذ بعد KeyDown و قبل KeyUp
KeyUp	هذا الحدث يتعلق بلوحة المفاتيح أيضا، ويتولد حينما نرفع إصبعنا عن المفتاح وهو طبعا يتولد بعد الحدثين KeyDown و KeyPress
GotFocus	يتولد هذا الحدث، حينما يصل التحديد Focus إلى الأداة المعنية.
LostFocus	يتولد هذا الحدث حينما تفقد الأداة التحديد

	Focus منتقلا إلى غيرها.
MouseDown	هذا الحدث شبيه بالحدث Click ولكنه لا يتطلب منك أن تترك زر الماوس بعد الضغط، فقط بمجرد الضغط على الأداة يتولد هذا الحدث.
MouseUp	هذا على عكس الحدث السابق، ينفذ بعد أن تترك الضغط بزر الماوس، كما لو أن مجموع هذين الحدثين يشكل الحدث Click
MouseMove	يتولد هذا الحدث عند مرور مؤشر الماوس فوق الأداة.
Resize	ينفذ هذا الحدث حينما يقوم المستخدم بتغيير مقاس الفورم.
Validated	ينفذ هذا الحدث بعد انتقال التحديد Focus من أداة إلى أخرى، ويستعمل غالبا في التحقق

من محتوى الأداة قبل المرور إلى الأداة الموالية.

Validating	له نفس دور الحدث السابق إلا أنه يتحقق من محتوى الأداة والتحديد Focus موجود عليها، لذا فهو ينفذ قبل الحدث Validated

آمل أن تكون قد أخذت فكرة عن مفهوم الخصائص والأحداث والأدوات، الآن سوف نمر إلى تفصيل بعض الأدوات التي ستحتاجها في برامجك، وسنسعى قدر المستطاع إلى تبسيط الشرح حتى تنهي هذا الفصل وأنت على ثقة بأنك قادر على إنجاز أي برنامج تريده في بيئة النوافذ.

الأدوات Controls

حتى نتفادى التكرار فلن نتطرق إلى شرح الأدوات مرة أخرى، بل سندخل مباشرة في شرح الأدوات، وأول أداة سنبدأ معها هي أداة الزر Button.

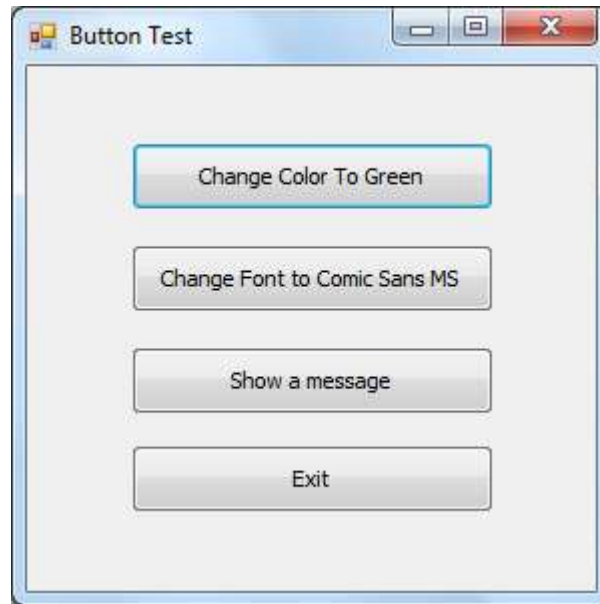
1. أداة الزر Button

رأينا شكله سابقا في أول تطبيق عملي، وهو عبارة عن أداة تستعمل للقيام بعمل ما عند الضغط عليها، تماما مثل زر المذياع الذي تضغط عليه للتشغيل، أو كزر الجرس الذي تضغط عليه ليبدأ الجهاز في الرنين، وتعد هذه الأداة واحدة من أهم الأدوات التي لا يخلو منها أي برنامج، ولك أن تجول في كل البرامج الموجودة على حاسوبك لتقف على صدق كلامي، فهي لازمة وضرورية.

This is a button

صورة لأداة الزر

بالنسبة لخصائص الأداة فقد ذكرنا أهمها في الجدول الجامع، وكذلك سردنا بعض أحداثها، وإن كان أكثر أحداث الزر استعمالا هو Click، سنقوم بعمل تطبيق بسيط لنستأنس بهذه الأداة.

افتح مشروعا جديدا، ولتسمه مثلا Button Test ، بعد ذلك ضع على الفورم أربعة أزرار، وقم بتغيير النصوص الظاهرة عليها كما يلي:

- الزر الأول يقوم بتغيير لون خلفية الفورم إلى الأخضر.
- الزر الثاني يقوم بتغير الخط الافتراضي إلى Comic Sans MS
- الزر الثالث يقوم بإظهار رسالة

الزر الرابع يقوم بإغلاق البرنامج

لكتابة الكود الخاص بالزر الأول، سنقوم بتحديده والذهاب إلى نافذة الخصائص ثم الضغط على أيقونة الأحداث واختيار الحدث Click ثم نضغط عليه مرتين، أو بكل بساطة نضغط مرتين على الزر.

ثم اكتب هذا السطر داخل الإجراء:

```
private void button1_Click(object sender, EventArgs e)
        {
            this.BackColor=System.Drawing.Color.Green;
        }
```

أو قم بجلب مجال الأسماء System.Drawing واختصر الكود، كما يلي:

```
using System;
using System.Windows.Forms;
using System.Drawing;
namespace ButtonTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.BackColor=Color.Green;
        }
    }
}
```

نفذ البرنامج واضغط على الزر الأول وشاهد كيف سيتحول لون الفورم من العادي إلى الأخضر.

الآن أدخل إلى الحدث Click الخاص بالزر الثاني، واكتب ما يلي:

```
private void button2_Click(object sender, EventArgs e)
        {
            this.Font = new Font("Comic Sans MS", 12);
        }
```

في البرامتر الأول للفئة Font نضع اسم الخط، وفي البرامتر الثاني نضع حجم الخط.

قم بتنفيذ البرنامج واضغط على الزر الثاني وشاهد كيف سيتغير نوع وحجم الخط.

ننتقل الآن إلى الزر الثالث الذي يقوم بإظهار رسالة، سأتركك تقوم بذلك لوحدك لأننا رأينا فيما سبق كيف نظهر رسالة عند الضغط على الزر.

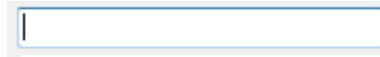
فيما يخص الزر الرابع الذي يقوم بإغلاق البرنامج، فيكفيك الولوج إلى الحدث Click الخاص به واكتب الإجراء التالي:

```
private void button4_Click(object sender, EventArgs e)
        {
            this.Close();
        }
```

ملحوظة:

العبارة this تعني الفورم الحالي الذي نشتغل عليه.

2. علبة النص TextBox

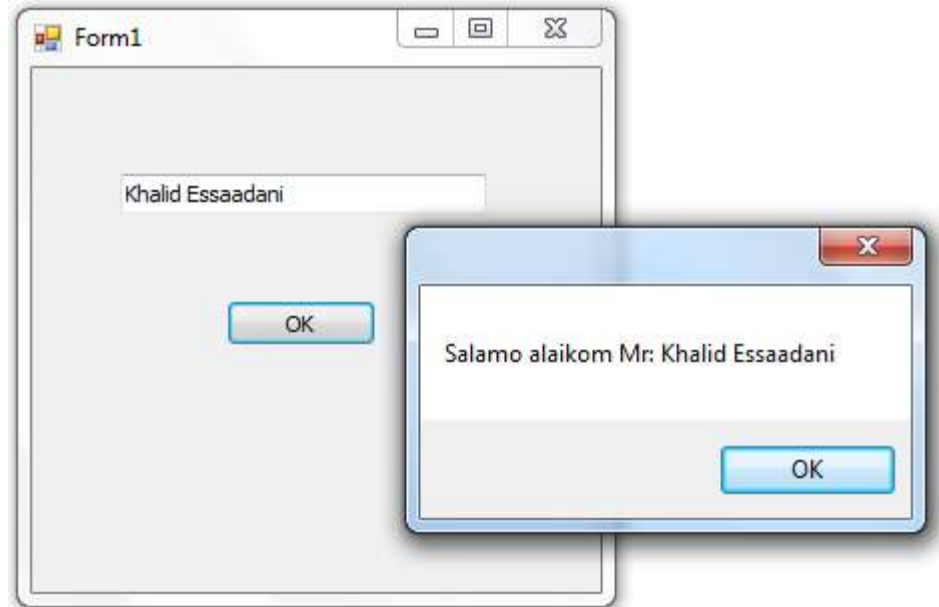
هذه الأداة لا تقل أهمية عن أداة Button، بحيث نجدها أيضا في أغلب البرامج، وهي عبارة عن أداة لإدخال النصوص، وكذلك لإظهارها، وأهم خاصية لهذه الأداة هي Text التي من خلالها نقرأ النص الذي يدخله المستخدم، وكذلك عبر نفس الخاصية نظهر ما نشاء من النصوص.

صورة لأداة علبة النص

يقدم الجدول التالي بعض خصائص هذه الأداة:

الخصيصة	دورها
TextAlign	لتحديد موضع النص على الأداة (يمين، شمال، أو في الوسط).
MaxLength	لتحديد عدد الأحرف المسموح بكتابتها داخل علبة النص.

MultiLine	للسماح بتعدد الأسطر في الأداة، لأنها افتراضيا تسمح بسطر واحد فقط.
ReadOnly	لجعل الأداة خاصة بإظهار البيانات فقط، بمعنى تمنع الكتابة فيها وتسمح للمستخدم بالقراءة فقط.
PasswordChar	لتحويل شكل النص إلى حروف مرمزة، وتستعمل هذه الخاصية حينما نريد كتابة كلمة المرور أو أي معلومات سرية.
CharacterCasing	لتحديد حالة الأحرف (كبيرة Upper أو صغيرة Lower أو عادية Normal)
RighToLeft	لتغيير توجيه النص، ليسمح بالكتابة انطلاقا من اليمين، هذه الخاصية إذا أردنا كتابة نصوص عربية أو بأي لغة تبدأ من اليمين.

الآن بعون الله سننجز تطبيقا آخر يضم علبة نص، وزر.

يقوم هذا التطبيق بالسماح للمستخدم بكتابة اسمه في علبة النص، ثم يظهر له رسالة ترحيبية بعد أن يضغط على الزر.

بعد أن تضيف الأداتين إلى الفورم، قم بتغيير بعض الخصائص كما تشاء لتتعرف عليها، وإذا أحببت قم بتغيير اسم علبة النص من textBox1 إلى textName عن طريق الخاصية Name

حينما تنتهي من ذلك، ادخل إلى الحدث Click الخاص بالزر واكتب السطر التالي:

```
private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Salamo alaikom Mr: " + textName.Text);
        }
```

العلامة + تقوم بالدمج بين قيمتين من نوع نصي، وقد رأينا ذلك في الجزء الأول.

نفذ البرنامج، واكتب اسمك في علبة النص، ثم اضغط على الزر وشاهد النتيجة.

3. علبة النص الغنية RichTextBox

هذه الأداة شبيهة بالأداة السابقة TextBox، كونها تستطيع إظهار وإدخال النص، إلا أنها تتوفر على مزايا إضافية تنعدم في الأولى، ولعل أبرز هذه المميزات هي إمكانية احتوائها على نص متعدد الألوان والخطوط والأحجام، الشيء الذي تفتقد إليه أداة TextBox.

وتستعمل هذه الأداة في برامج معالجة النصوص كبرنامج الوورد Microsoft Word الشهير، وهذه صورة للأداة وهي تحتوي على ثلاثة أسطر وكل سطر يخالف الأسطر الأخرى في لونه وحجمه ونوع خطه:

تستطيع تغيير لون الخط المحدد كما يعرض هذا السطر:

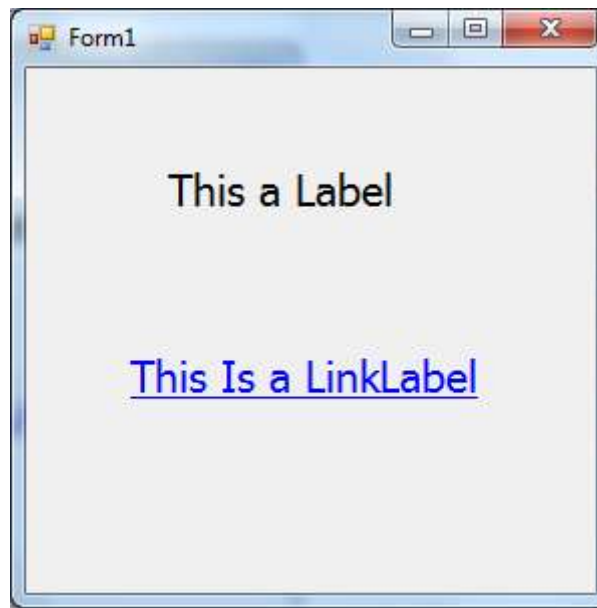
```
this.richTextBox1.SelectionColor = Color.Red;
```

ولتغيير نوع و حجم الخط انظر إلى السطر التالي:

```
this.richTextBox1.SelectionFont = new Font("Times New Roman", 16);
```

بحيث تضع في البرامتر الأول اسم الخط بين مزدوجتين، وفي البرامتر الثاني حجم النص.

4. أدوات إظهار النص LinkLabel و Label

وهي أداة تلعب دور الملصق الذي يكون مطبوعا على الملابس أو على الأجهزة بغرض تعريفها، و دورها إظهار النص فقط ، هذا في حالة الأداة Label، أما في حالة الأداة LinkLabel فهي تلعب نفس الدور إضافة إلى كونها عبارة عن رابط أنترنت HyperLink

لتحديد رابط للأداة LinkLabel، اذهب إلى الحدث LinkClicked الخاص بها، أو اضغط عليها مرتين واكتب هذا السطر الذي يقوم بفتح أي رابط أنترنت تريده:

```
        private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            System.Diagnostics.Process.Start("www.google.com");
        }
```

أو قم بجلب مجال الأسماء System.Diagnostics لتختصر الكتابة كما يلي:

```
using System;
using System.Windows.Forms;
using System.Diagnostics;
namespace LinkLabelTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void linkLabel1_LinkClicked(object sender,
LinkLabelLinkClickedEventArgs e)
        {
            Process.Start("www.google.com");
        }
    }
}
```

5. علبة الاختيار CheckBox

وهذه صورة الأداة قبل أن نتحدث عنها:

كما ترى فإن هذه الأداة تسمح للمستخدم باختيار متعدد، ولعلك صادفتها في أكثر من برنامج، هذه الأداة تقبل حالتين لا ثالث لهما، ويمكن إضافة حالة ثالثة إذا قمنا بتغيير الخصيصة ThreeState إلى True ، ولكن عموما فسوف تتعامل فقط مع هاتين الحالتين:

- **محددة** Checked: حينما تقوم بعملية الاختيار تصبح حالة الأداة محددة.
- **غير محددة** UnChecked: حينما لا تختار أو تلغي الإختيار تصبح حالة الأداة هكذا.

قم بإنجاز الفورم التالي:

سنحاول من خلال هذا التطبيق البسيط إظهار اختيار المستخدم في رسالة حينما يضغط على الزر، طيب قم بالدخول إلى الحدث Click الخاص بالزر، وقم بكتابة الشفرة التالية، قبل ذلك قم بإعادة تسمية الأداتين عبر الخصيصة Name، وليكن اسم علبة التحديد الأولى CheckVB، وليكن اسم الثانية CheckCS:

```
        private void button1_Click(object sender, EventArgs e)
        {
            //إذا حدد المستخدم الخيار الأول
            if (CheckVB.Checked == true && CheckCS.Checked == false)
            {
                MessageBox.Show("Your Favourite language is : Visual Basic.net");
            }
            //إذا حدد المستخدم الخيار الثاني
            else if (CheckVB.Checked == false && CheckCS.Checked == true)
            {
                MessageBox.Show("Your Favourite language is : CSharp.net");
            }
            //إذا لم يحدد المستخدم أي خيار
            else if (CheckVB.Checked == false && CheckCS.Checked == false)
            {
                MessageBox.Show("Please Choose Your language !");
            }
            //إذا حدد المستخدم الخيارين معا
            else
            {
                MessageBox.Show("Your Favourite language is : Visual Basic.net
and CSharp.Net");
            }
        }
```

6. أداة زر الاختيار RadioButton

وهذه صورة لهذه الأداة قبل أن ننطلق في الحديث عنها:

ستبدو لك هذه الأداة مشابهة لعلبة الاختيار CheckBox، معك الحق فهما يشتركان تقريبا في الخصائص، ولكن أحيطك علما بأن هذه الأداة لا تتيح للمستخدم إمكانية تعدد الاختيارات، بحيث يصير له الحق في اختيار واحد.

أنجز الفورم أعلاه، وأضف إليه زرا لعرض اختيار المستخدم، كما يلي:

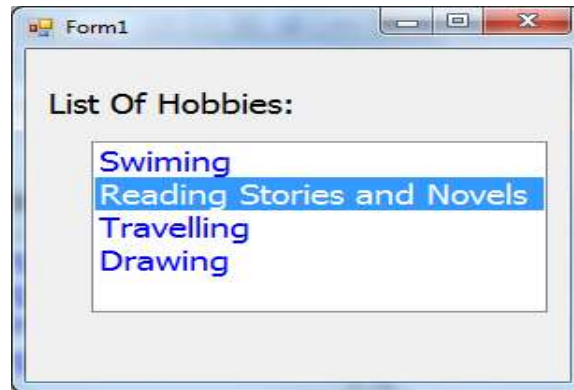
ثم قم بإعادة تسمية زري الاختيار، وليكن مثلا (RBsingle، RBmarried)

بعد ذلك، ادخل إلى الحدث Click للزر، ثم اكتب ما يلي:

```
private void button1_Click(object sender, EventArgs e)
        {
            if (RBsingle.Checked == true)
            {
                MessageBox.Show("You are Single ");
            }
            else
            {
                MessageBox.Show("You are Married ");
            }
        }
```

أعتقد بأن الأمر لا يحتاج إلى شرح، أنجز تطبيقا خاصا بك، تستعمل فيه أزرار الاختيار بطريقة مغايرة لهذا المثال وبرهن لي بأنك أسد 😉

7. علبة القائمة ListBox

انظر إلى الأداة أولا:

تقوم هذه الأداة بعرض البيانات على شكل قائمة، يمكن للمستخدم من خلالها أن يختار ما يشاء، ويمكنك تعبئتها يدويا عن طريق الذهاب إلى الخصيصة Items، وملؤها بما تشاء، أما إذا أردت أن تملأها بواسطة الكود فعليك بالدخول إلى الحدث Load الخاص بالفورم، إما عن طريق الذهاب إلى نافذة الخصائص واختيار الحدث بعد تحديد أيقونة الأحداث، أو بكل سهولة عن طريق عمل ضغطتين متتاليتين Double Click على الفورم، بعد ذلك اكتب ما يلي:

```
private void Form1_Load(object sender, EventArgs e)
    {
        this.listBox1.Items.Add("First Item");
        this.listBox1.Items.Add("Second Item");
        this.listBox1.Items.Add("Third Item");
    }
```

ملحوظة:

يمكنك أيضا تعبئة القائمة داخل مشيد الفئة Form1 بعد الدالة initialize()، لأن المشيد هو أول ماينفذ:

```
public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.listBox1.Items.Add("First Item");
            this.listBox1.Items.Add("Second Item");
            this.listBox1.Items.Add("Third Item");
        }
    }
```

أنجز برنامجا يتكون من علبة نص TextBox و زر Button وعلبة القائمة ListBox وأداة إظهار النص Label، يكون شكله كما يلي:

سيقوم المستخدم بكتابة اسمه، ثم يضغط على الزر Add To list لينضاف اسمه إلى القائمة، إذا أراد المستخدم أن يحذف سطرا معينا ما عليه سوى تحديده ثم الضغط على الزر السفلي.

الآن ادخل إلى الحدث Click الخاص بالزر Add To list واكتب مايلي:

```
private void button1_Click(object sender, EventArgs e)
{
    this.listBox1.Items.Add(textBox1.Text);
}
```

الدالة Add تقوم بإضافة الاسم المكتوب في علبة النص إلى القائمة.

الآن اذهب إلى زر الحذف واكتب مايلي:

```
private void button2_Click(object sender, EventArgs e)
{
    this.listBox1.Items.RemoveAt(listBox1.SelectedIndex);
}
```

الدالة RemoveAt تقوم بحذف السطر من خلال رتبته في القائمة، لهذا كتبنا داخل البرامتر listBox1.SelectedIndex لأن الخاصية SelectedIndex تعيد لنا رتبة السطر المحدد.

لحذف كل الأسطر دفعة واحدة استعمل الدالة Clear:

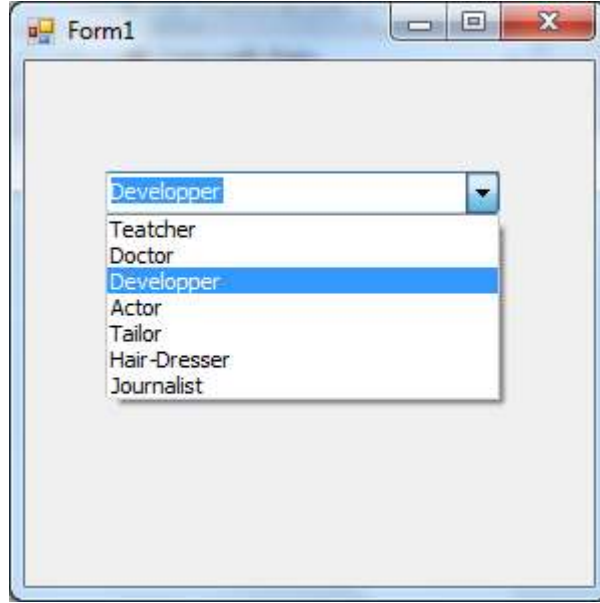
```
this.listBox1.Items.Clear();
```

الآن سنقوم بتعبئة الأداة بواسطة مصفوفة، إن نسيت معنى المصفوفات Arrays عد إلى الجزء الأول:

```
namespace ListBoxFromArrays
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            string[] WeekDay = { "Sunday", "Monday", "Tuesday",
"Wednesday", "Thursday", "Friday", "Saturday" };

            this.listBox1.Items.AddRange(WeekDay);
        }
    }
}
```

الدالة AddRang تقوم بإضافة جدول إلى ListBox، أنجز مثالا من مخيلتك لتستأنس بهذه الأداة.

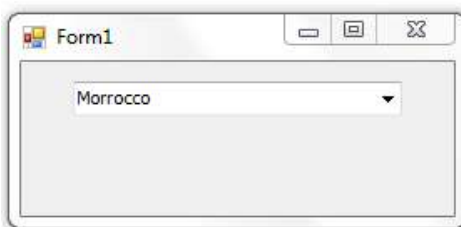
8. علبة الكومبو ComboBox

لا شك أنك شاهدت مثل هذه الأداة في أكثر من برنامج، وأكثر من موقع انترنت:

هذه هي أداة علبة الكومبو، والتي تستعمل لتمكين المستخدم من اختيار بعض البيانات، مثلا حينما تقوم بالتسجيل في موقع الفيسبوك أو أي موقع آخر، فإنه يطلب منك أن تختار الدولة التي تنتمي إليها.

بالنسبة لخصائص وأحداث هذه الأداة فهي شبيهة بخصائص وأحداث أداة علبة القائمة ListBox، سنقوم بتطبيق بسيط من باب الاستئناس بهذه الأداة، قم بإنشاء مشروع جديد، وضع على الفورم أداة علبة الكومبو، ثم املأها بأسماء الدول، إما يدويا عن طريق الذهاب إلى الخصيصة Items، أو عن طريق الشفرة بواسطة الدالة AddRange

```
namespace ComboboxTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            string[] Countries =
            {"Egypt","Morrocco","Algeria","Iraq","Tunisia","Libya"};
            this.comboBox1.Items.AddRange(Countries);
        }
    }
}
```

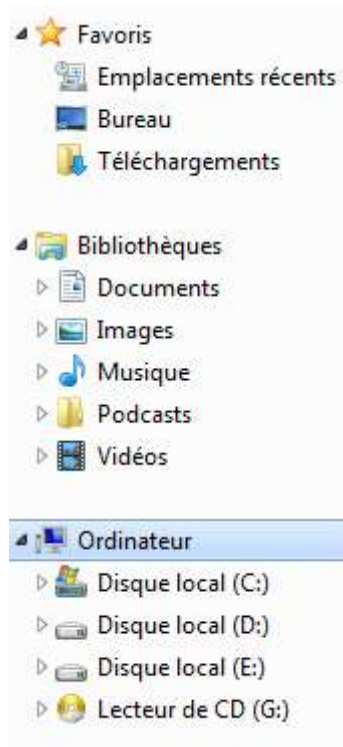
نريد أن نظهر رسالة للمستخدم تظهر له اسم الدولة التي اختارها، بعد أن يقوم بتحديد دولته، الحدث الذي يتولد عند الاختيار من علبة الكومبو هو SelectedIndexChanged ، للولوج إليه نضغط مرتين على أداة الكومبو ونكتب السطر التالي:

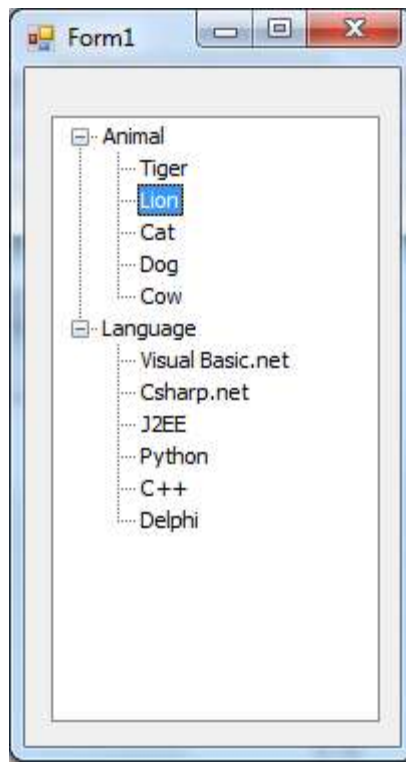
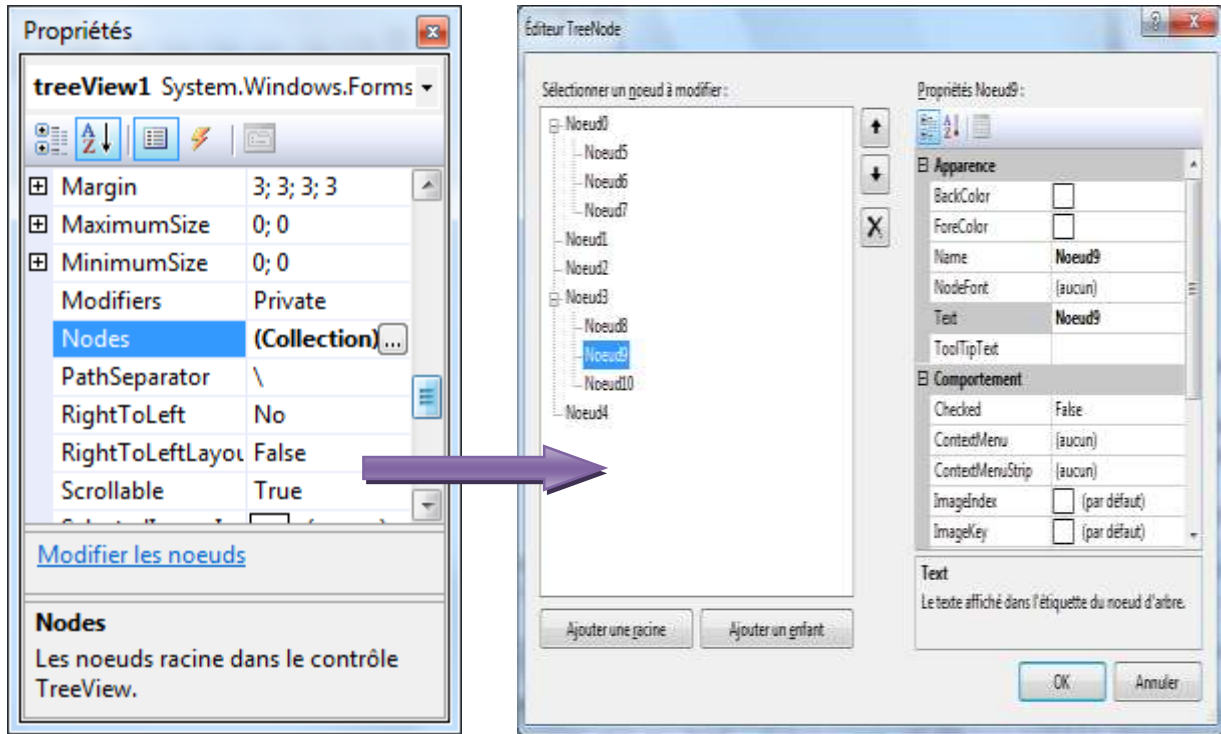
```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            MessageBox.Show("Your Country is :  " + comboBox1.Text);
        }
```

قم بالتنفيذ، وشاهد النتيجة:

هنالك خصيصة تقوم بتغيير طريقة عرض البيانات داخل الكومبو، وهي DropDownList، وتأخذ القيم الثلاثة التالية:

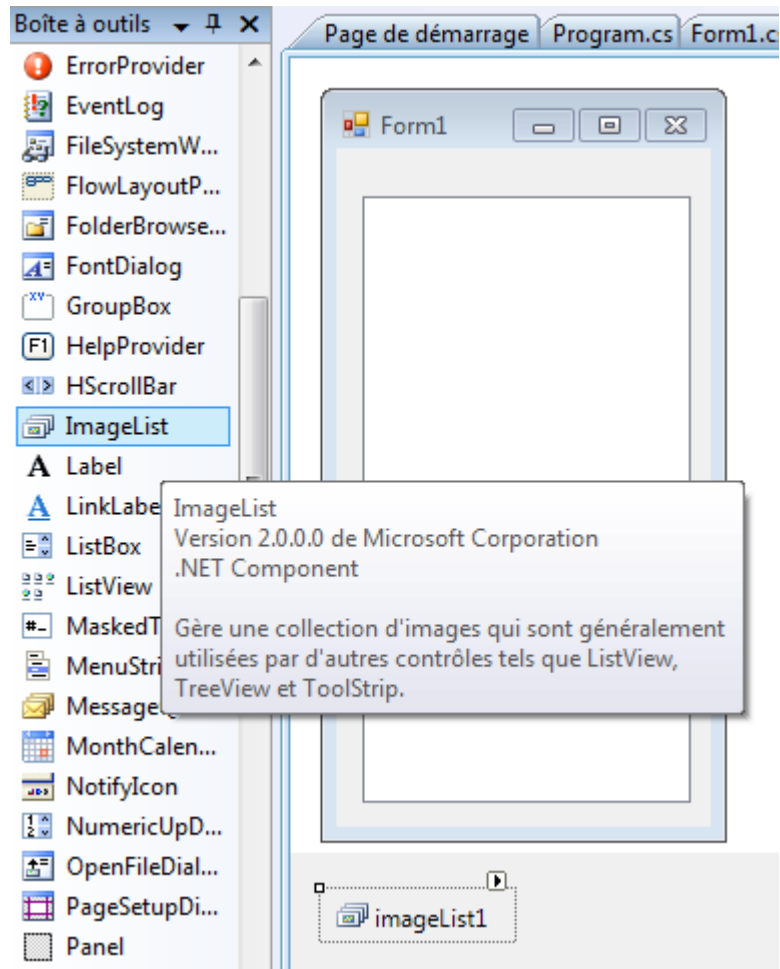
- Simple: تجعل الكومبو شبيها بعلبة النص.
- DropDownList: تظهر البيانات على شكل قائمة غير قابلة للتعديل.
- DropDown: هذا هو الوضع العادي لأداة الكومبو.

9. القائمة الشجرية TreeView

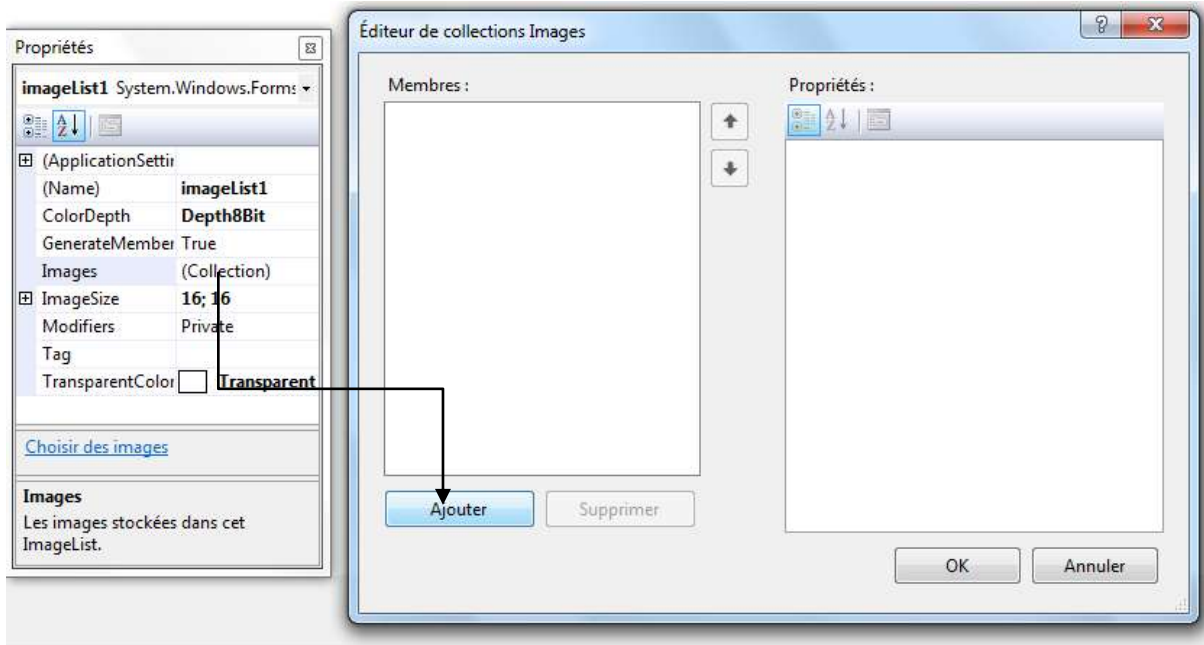
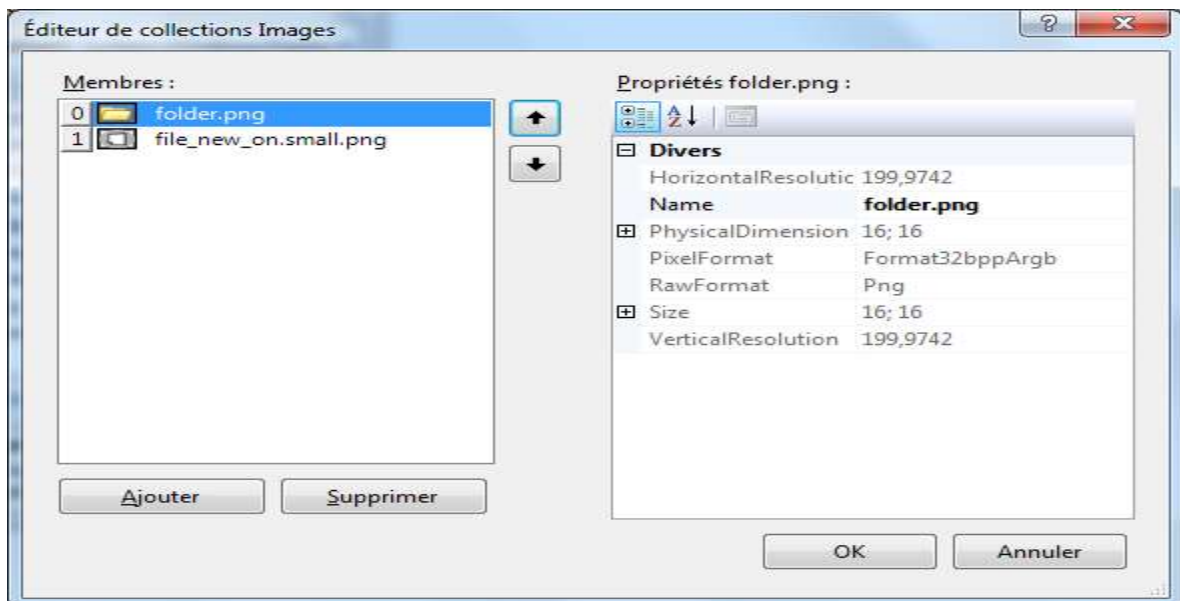
طبعا سبق لك أن رأيتها في متصفح الويندوز:

وهي أداة تقوم بإظهار البيانات تنازليا على شكل بينة شجرية متشعبة، من الأصل إلى الفروع، وتستعمل غالبا في تصفح وحدات الحاسوب Drives، وكذلك الملفات Files

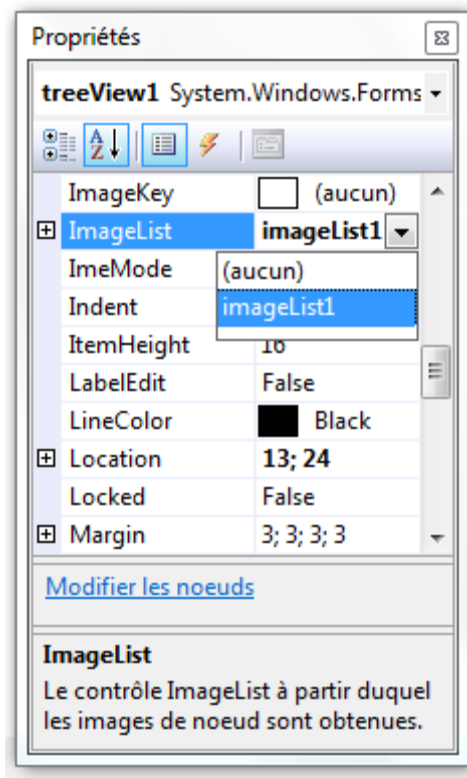
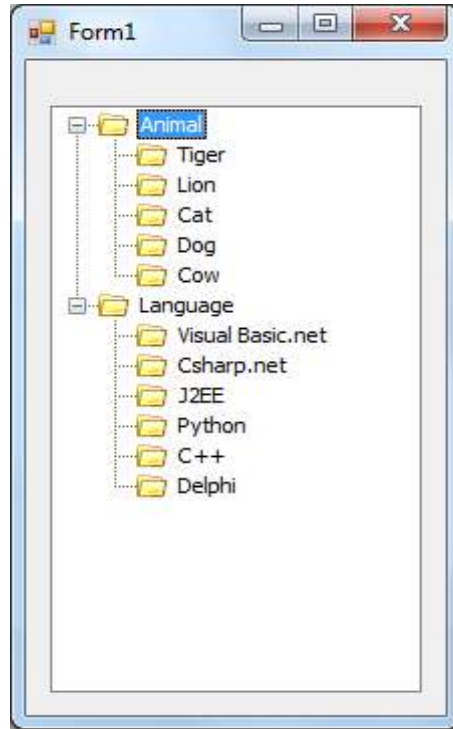
والمجلدات Folders، سنقوم بإنجاز مثال يعرض مجموعات، وداخل كل مجموعة يبسط عناصرها، كما تظهر الصورة التالية:

بإمكاننا عمل ذلك يدويا بكل سهولة، عن طريق نافذة الخصائص بواسطة الخصيصة Nodes، كما يبدو جليا هنا:

كما تلاحظ فيمكننا من خلال نافذة الخصائص إضافة صور إلى أداة الشجرة، وتغيير الألوان والخطوط، أما فيما يخص عمل ذلك بواسطة الشفرة فذلك سهل أيضا، كل ما في الأمر هو إضافة العقدة الأم Node، ثم إضافة العقد التفصيلية من خلال رتبة العقدة الأم، بنفس الطريقة نتحكم في الخصائص أيضا، سنورد الآن كيفية إنشاء النموذج السابق بواسطة الشفرة، ثم سنرى إن شاء الله كيف نتحكم في مظهر العقد Nodes. لاحظ معي جيدا كيف عملنا ذلك بواسطة الكود:

```
namespace TreeViewTest
{
    public partial class Form1 : Form
    {

        public Form1()
        {

            InitializeComponent();

            //إضافة العقدة الأولى وتفريعاتها
            treeView1.Nodes.Add("Animal");
            treeView1.Nodes[0].Nodes.Add("Tiger");
            treeView1.Nodes[0].Nodes.Add("Lion");
            treeView1.Nodes[0].Nodes.Add("Cat");
            treeView1.Nodes[0].Nodes.Add("Dog");
            treeView1.Nodes[0].Nodes.Add("Cow");


            //إضافة العقدة الثانية وتفريعاتها
            treeView1.Nodes.Add("Language");
            treeView1.Nodes[1].Nodes.Add("Visual Basic.net");
            treeView1.Nodes[1].Nodes.Add("Csharp.net");
            treeView1.Nodes[1].Nodes.Add("J2EE");
            treeView1.Nodes[1].Nodes.Add("Python");
            treeView1.Nodes[1].Nodes.Add("C++");
            treeView1.Nodes[1].Nodes.Add("Delphi");

        }

    }
}
```

الآن سنقوم بإضافة الصور إلى الأداة ليبدو مظهرها أكثر احترافية، أولا قم بالذهاب إلى علبة الأدوات، واجذب أداة ImageList إلى الفورم:

10. أداة قائمة الصور ImageList

هذه الأداة تعتبر خزانا لحفظ الصور، ليتم استعمالها من قبل أداة أخرى.

قم بتحديد الأداة واذهب إلى خصائصها وبالضبط إلى الخصيصة Images، ثم قم بوضع الصور التي تريد تضمينها لأداة الشجرة TreeView

من خلال زر "أضف"، قم بجلب الصور إلى أداة ImageList.

حينما تنتهي من جلب الصور، اضغط على الزر OK.

الآن قمنا بجلب الصور وتخزينها في أداة ImageList، بقي لنا فقط ربط هذه الأداة مع أداة TreeView، الأمر بسيط جدا، قم بتحديد أداة Treeview ثم اذهب إلى خصائصها وتحديدا إلى الخصيصة ImageList، ومن خلالها قم باختيار أداة ImageList1 التي أضفنا إليها الصور:

إذا نفذت البرنامج الآن، ستلاحظ تغير صور العقد، ولكن صورة واحدة هي التي تظهر.

لتعديل ذلك، سنضيف بعض التعديلات التي تقوم بترتيب الصور حسب العقد، حيث ستأخذ العقد الأم الصورة الأولى من ImageList1، بينما تأخذ العقد المتفرعة الصورة الثانية من ImageList1، للإشارة فقط فترتيب الصور يبدأ من 0، بمعنى أن ترتيب الصورة الثانية هو 1.

كما رأينا فهذا السطر يقوم بإضافة عقدة إلى الشجرة TreeView:

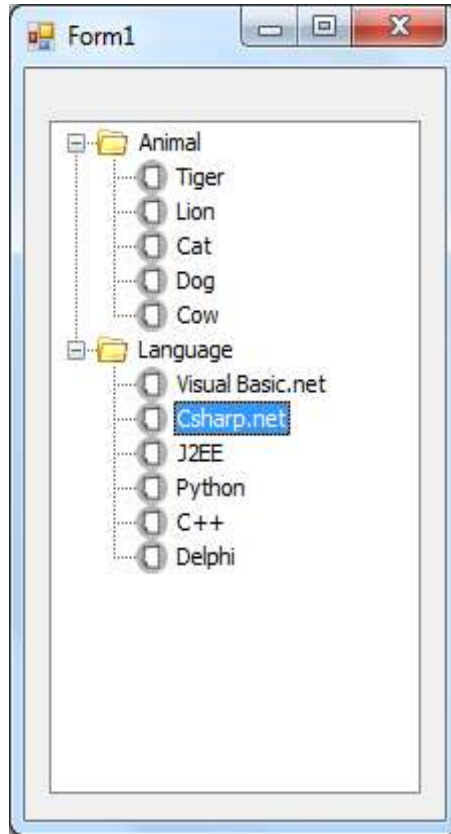
```
treeView1.Nodes.Add("Animal");
```

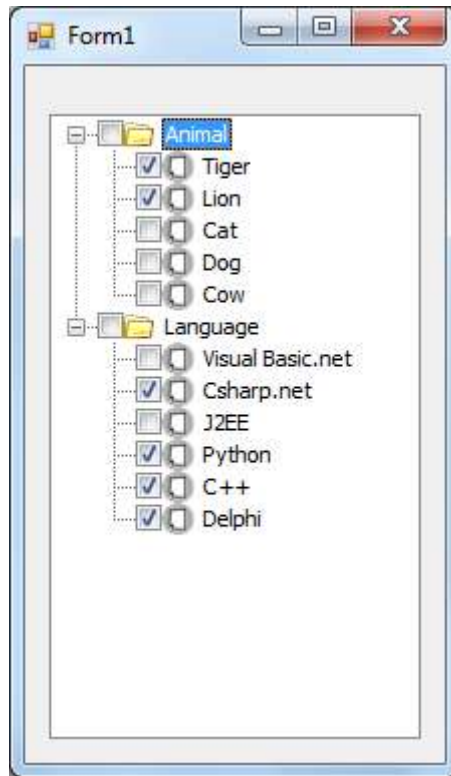
الدالة Add تقبل العديد من البرامترات، التي من بينها ترتيب الصورة، واسم العقدة وقيمة العقدة، وكذلك صورة العقدة عند تحديدها. أي كما يلي:

```
treeView1.Nodes.Add("اسم العقدة","قيمة العقدة",ترتيب الصورة,ترتيب الصورة عند التحديد);
```

سنقوم فقط بتعديل الكود السابق، مع تحديد رتبة الصورة 0، للعقد الرئيسية، والرتبة 1 للعقد الفرعية:

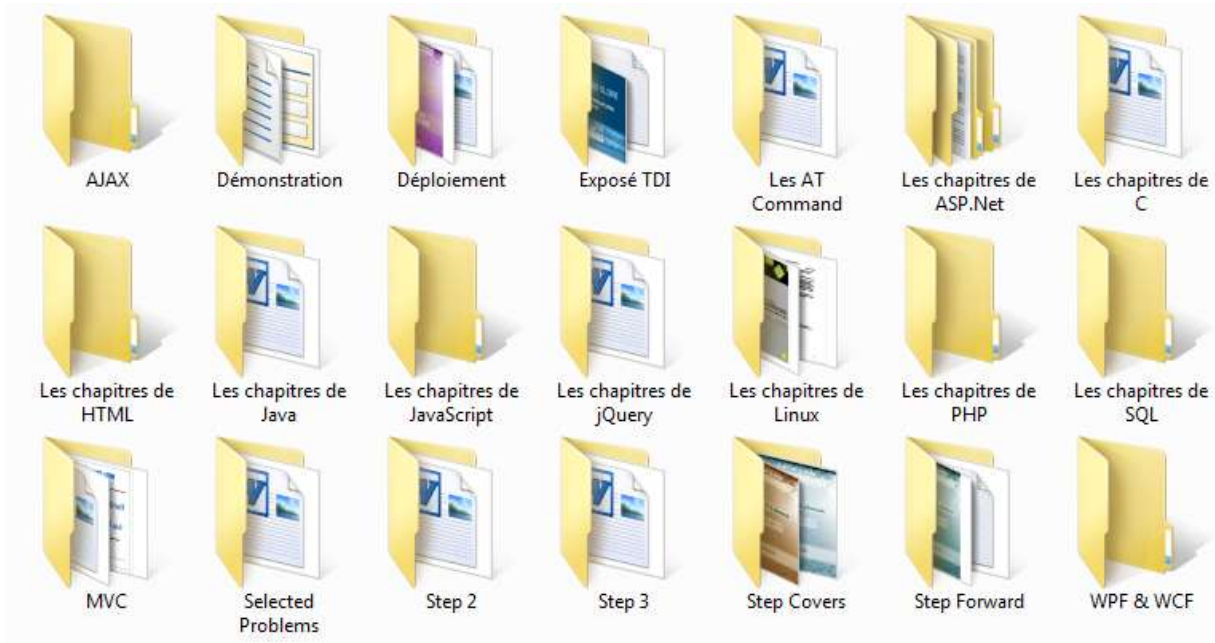
```
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            //إضافة العقدة الأولى وتفريعاتها
            treeView1.Nodes.Add("","Animal",0,0);
            treeView1.Nodes[0].Nodes.Add("", "Tiger", 1, 1);
            treeView1.Nodes[0].Nodes.Add("", "Lion", 1, 1);
            treeView1.Nodes[0].Nodes.Add("", "Cat", 1, 1);
            treeView1.Nodes[0].Nodes.Add("", "Dog", 1, 1);
            treeView1.Nodes[0].Nodes.Add("", "Cow", 1, 1);

            //إضافة العقدة الثانية وتفريعاتها
            treeView1.Nodes.Add("", "Language", 0,0);
            treeView1.Nodes[1].Nodes.Add("", "Visual Basic.net", 1, 1);
            treeView1.Nodes[1].Nodes.Add("", "Csharp.net", 1, 1);
            treeView1.Nodes[1].Nodes.Add("", "J2EE", 1, 1);
            treeView1.Nodes[1].Nodes.Add("", "Python", 1, 1);
            treeView1.Nodes[1].Nodes.Add("", "C++", 1, 1);
            treeView1.Nodes[1].Nodes.Add("", "Delphi", 1, 1);

        }
    }
```

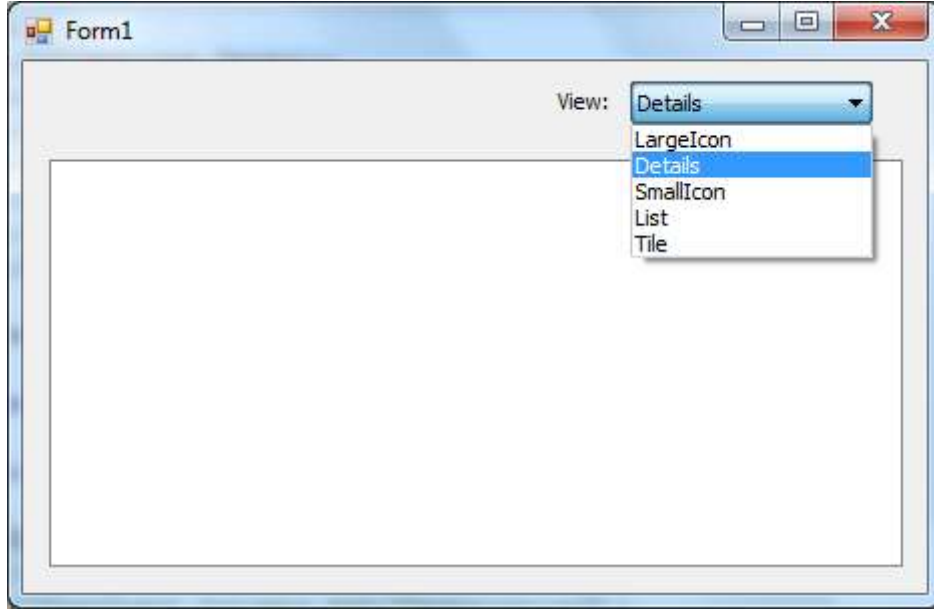
قم بحفظ التطبيق، ثم اضغط على F5 للتنفيذ وشاهد النتيجة:

أعتقد أن العقد قد عقدتك تعقيدا ، لا عليك فإنها لأول وهلة تبدو هكذا لكن بمجرد مراجعة ما سبق وتطبيقه ستصبح هكذا !!!

تتوفر أداة TreeView على خصيصة تسمى CheckBoxes، إذا غيرت قيمتها من False إلى True، سوف يظهر شكل الأداة كما يلي:

لتعرف العقد المحددة Checked، استعمل الشفرة التالية:

```
foreach (TreeNode Node in this.treeView1.Nodes)
{
    if (Node.Checked == true)
    {
        //Do Something :)
    }
}
```

11. قائمة العرض ListView

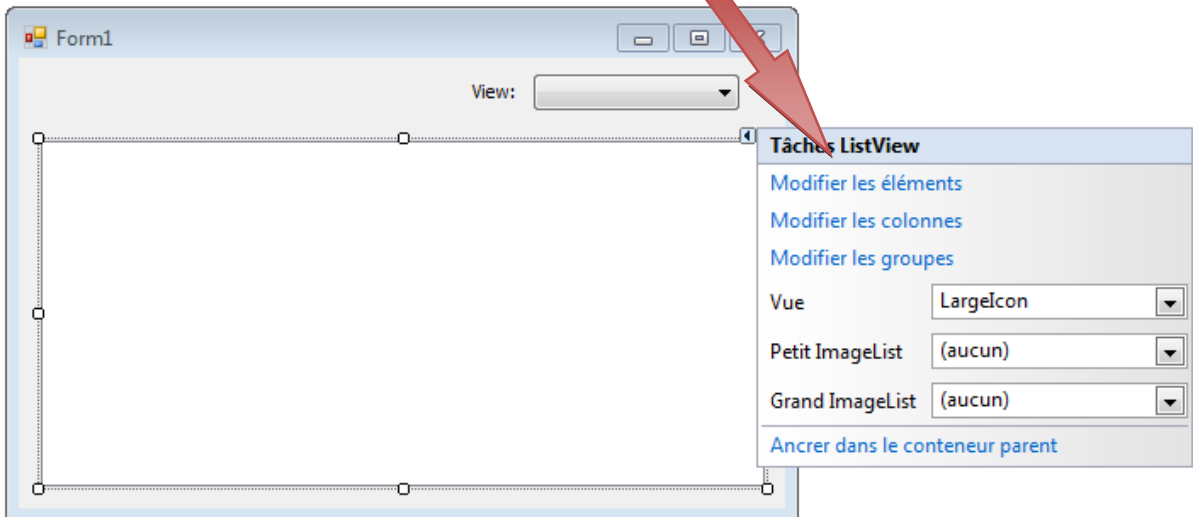
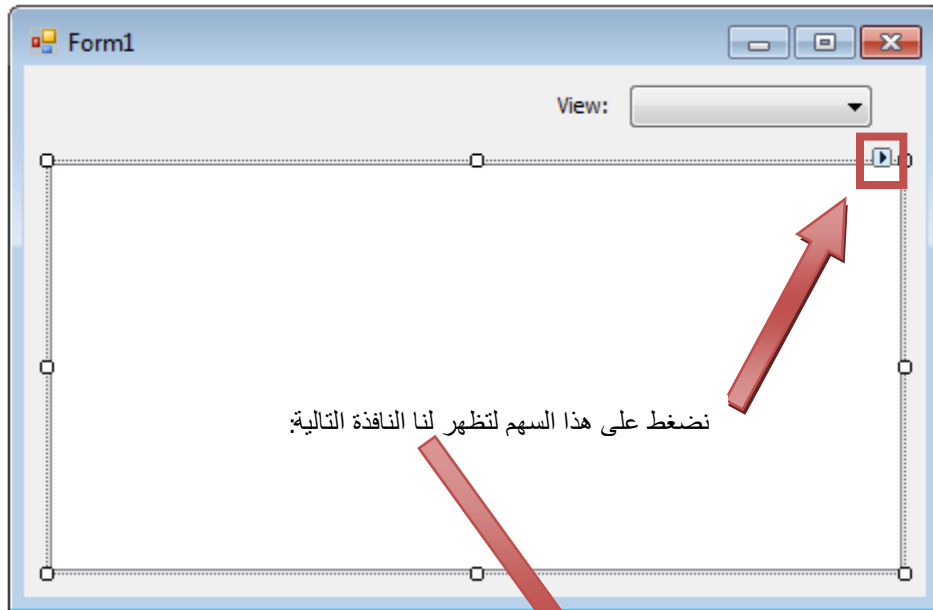
هذه الأداة شبيهة بأداة علبة القائمة ListBox، إلا أنها أكثر احترافية وجمالا منها، بيد أنها تستطيع عرض البيانات بطريقة منسقة وجذابة، كما تستطيع عرض الصور، وحتى تتعرف عليها أكثر، انظر إلى محتوى متصفح الويندوز واعلم أنه مصمم بهذه الأداة :

وتمكنك هذه الأداة من تغيير طريقة العرض كما يحلو لك.

سوف نقوم إن شاء الله الآن بإنشاء متصفح شبيه بمتصفح الويندوز Windows Explorer، لذا قم بفتح مشروع جديد، وأضف إلى الفورم الأدوات الظاهرة فيما يلي:

كما تلاحظ قمنا بتعبئة أداة الكومبو بالبيانات الواردة أعلاه، وقمنا بتغيير الخصيصة DropDownStyle إلى DropDownList، لنمنع المستخدم من الكتابة في أداة الكومبو.

في هذا البرنامج، سنظهر مجموعة من الأيقونات في أداة ListView، ونغير طريقة عرضها حينما يختار المستخدم من أداة الكومبو شكل العرض.

أولا قم بإضافة أداة ImageList إلى الفورم، لنضع فيها صور الأيقونات المراد عرضها، ثم بعد ذلك أضف إليها الصور بنفس الطريقة التي رأيناها سابقا.

حينما تضيف الصور إلى أداة ImageList1، قم بالربط بينها وبين أداة ListView، عن طريق الذهاب إلى السهم الظاهر أعلى أداة ListView، لكي تظهر لك النافذة التالية:

نضغط على هذا السهم لتظهر لنا النافذة التالية:

الآن لنشرح محتوى هذه النافذة الصغيرة:

- تعديل العناصر Edit Items: إذا أردنا إضافة عناصر إلى ListView يدويا.
- تعديل الأعمدة Edit Columns: إذا أردنا إضافة أعمدة جديدة أو تعديل أعمدة موجودة.
- تعديل المجموعات Edit Groups: إذا أردنا إضافة أو تغيير عرض البيانات عبر مجموعات.
- العرض View: للتحكم في طريقة عرض البيانات إما على شكل List أو Tile...
- مصدر الصور الصغيرة Small ImageList: مصدر الصور حينما نختار طريقة عرض البيانات على شكل صور مصغرة.
- مصدر الصور الكبيرة Large ImageList: مصدر الصور حينما نختار طريقة عرض البيانات على شكل صور مكبرة.

من خلال هذه النافذة سنحدد الخيارات التالية:

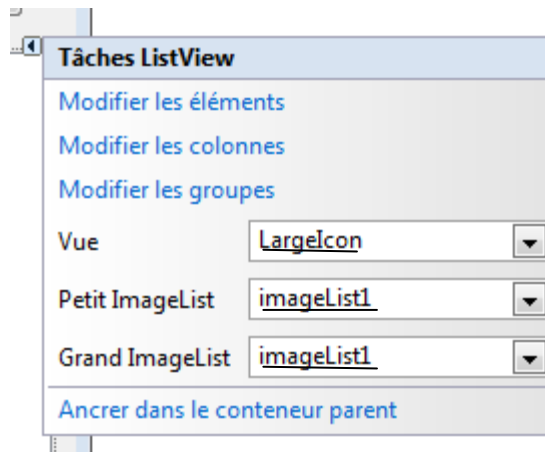
Tâches ListView

Modifier les éléments

Modifier les colonnes

Modifier les groupes

Vue	LargeIcon
Petit ImageList	imageList1
Grand ImageList	imageList1

Ancrer dans le conteneur parent

بعد أن أضفت الصور إلى أداة ImageList1، وغيرت الخصائص أعلاه ادخل إلى نافذة الكود، وأضف بعض العناصر إلى أداة Listview:

```
namespace TreeViewTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            listView1.Items.Add("Folder 1", 0);
            listView1.Items.Add("Folder 2", 0);
            listView1.Items.Add("Folder 3", 0);
            listView1.Items.Add("Folder 4", 0);
            listView1.Items.Add("Folder 5", 0);
        }
    }
}
```

البرامتر الأول للدالة Add هو اسم العنصر، والبرامتر الثاني هو رتبة الصورة المراد إظهارها للعنصر.

بعد أن تنفذ ستظهر النافذة بالشكل التالي:

الآن سنذهب إلى أداة الكومبو ونضغط عليها مرتين لكتابة شفرة تغيير العرض:

```
        private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (comboBox1.Text == "LargeIcon")
            {
                listView1.View = View.LargeIcon;
            }
            else if (comboBox1.Text == "Details")
            {
                listView1.View = View.Details;
            }
            else if (comboBox1.Text == "SmallIcon")
            {
                listView1.View = View.SmallIcon;
            }
            else if (comboBox1.Text == "List")
            {
                listView1.View = View.List;
            }
            else
            {
                listView1.View = View.Tile;
            }
        }
```

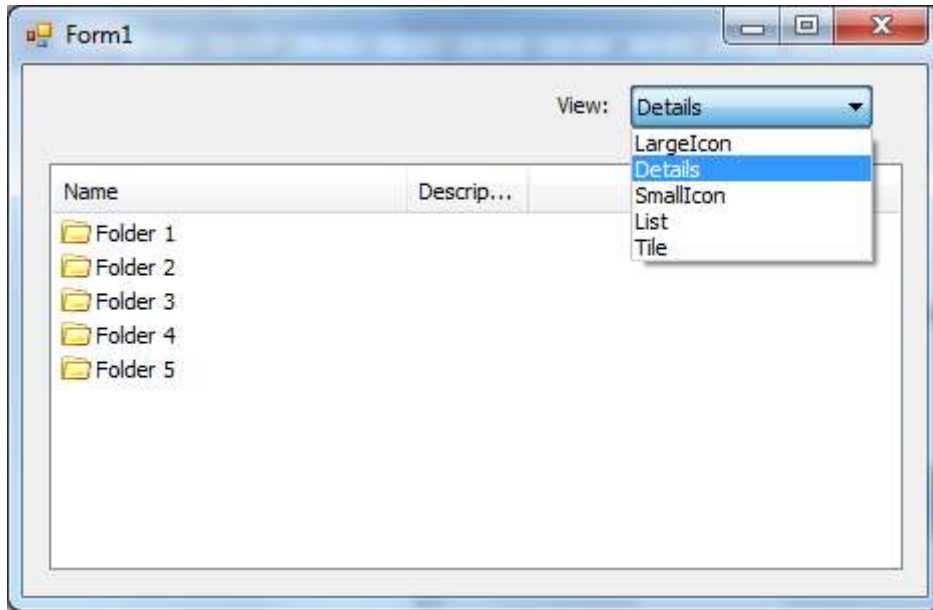
نقوم في هذه الشفرة من التحقق من اختيار المستخدم، ثم نغير طريقة العرض حسب ذلك.

جرب الآن ونفذ، سترى بأن طريقة العرض تتغير في جميع الاختيارات، ما عدا طريقة العرض Details، فستظهر أداة ListView فارغة المحتوى، الأمر منطقي، لأن هذه الطريقة في العرض تتطلب وجود رأس العمود Column Header، كما يبدو جليا هنا في متصفح الويندوز:

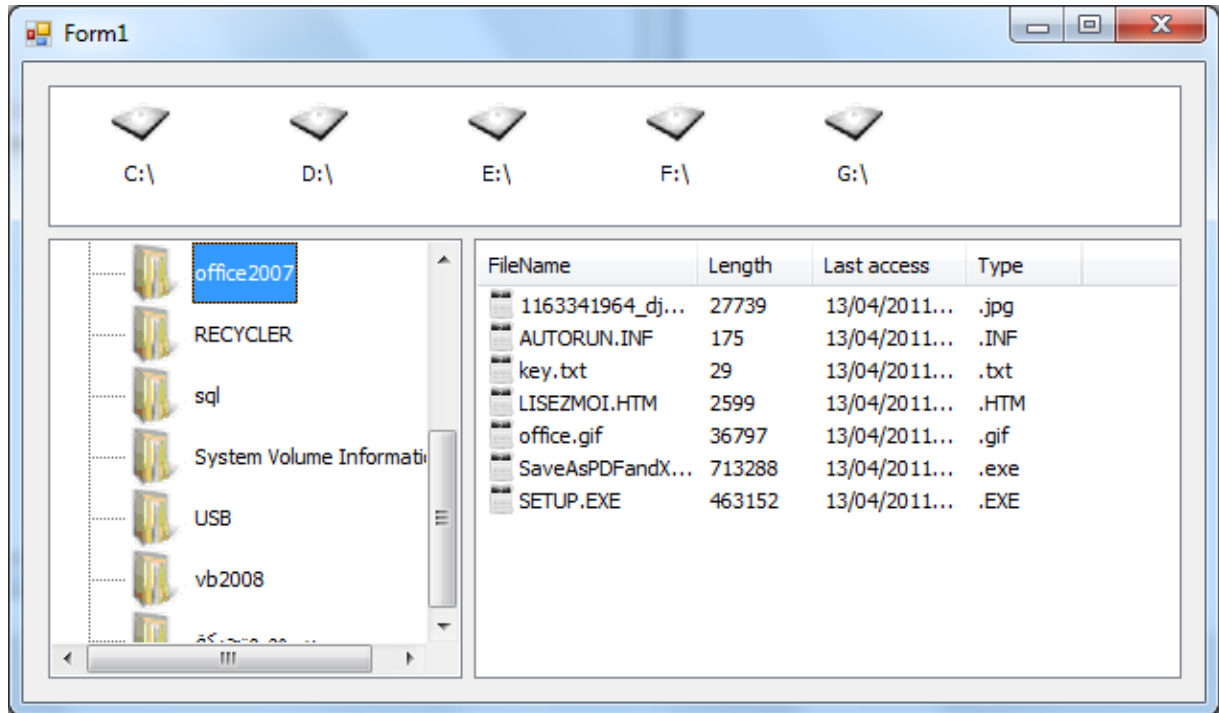
Nom	Modifié le	Type
AJAX	04/01/1980 00:10	Dossier de fichiers
Démonstration	22/09/2012 21:14	Dossier de fichiers
Déploiement	22/09/2012 21:14	Dossier de fichiers
Exposé TDI	22/09/2012 21:14	Dossier de fichiers
Les AT Command	22/09/2012 21:13	Dossier de fichiers
Les chapitres de ASP.Net	24/10/2012 01:17	Dossier de fichiers
Les chapitres de C	22/09/2012 21:13	Dossier de fichiers
Les chapitres de CPlus Plus	09/12/2012 13:15	Dossier de fichiers
Les chapitres de CSharp	05/12/2012 22:06	Dossier de fichiers
Les chapitres de CSS	04/01/1980 00:10	Dossier de fichiers
Les chapitres de HTML	04/01/1980 00:10	Dossier de fichiers
Les chapitres de Java	22/09/2012 21:13	Dossier de fichiers
Les chapitres de JavaScript	04/01/1980 00:10	Dossier de fichiers
Les chapitres de jQuery	22/09/2012 21:13	Dossier de fichiers
Les chapitres de Linux	07/11/2012 00:27	Dossier de fichiers

إذن سنقوم بإضافة الأعمدة إما يدويا عن طريق الأمر Edit Columns، الذي رأيناه في مستهل شرحنا لهذه الأداة، أو عن طريق الشفرة كما يلي:

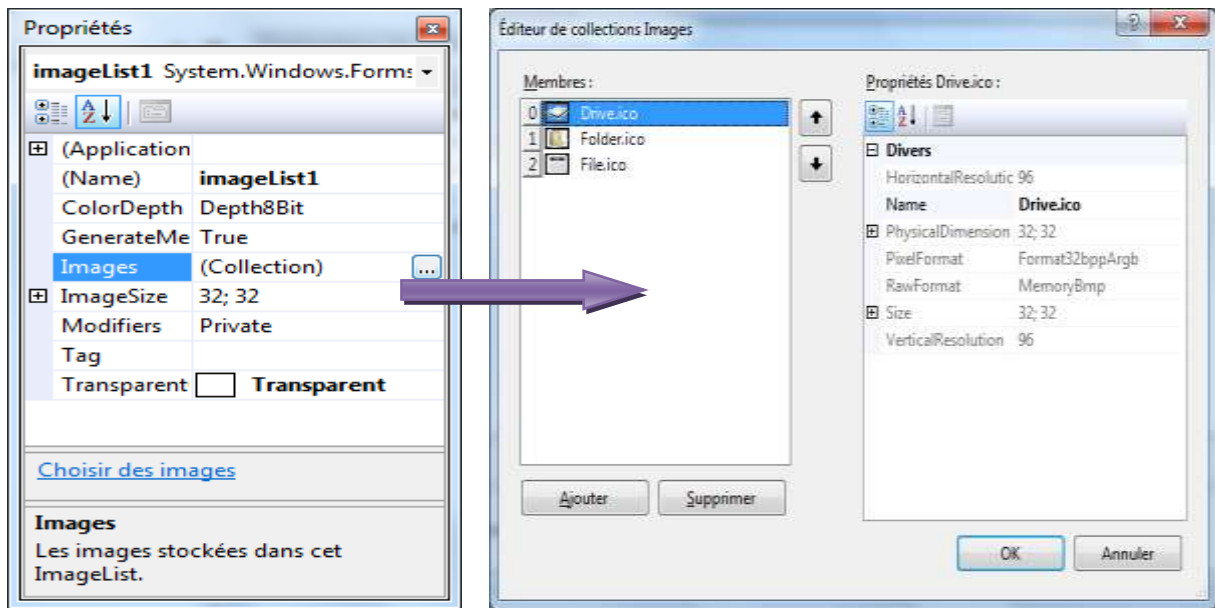
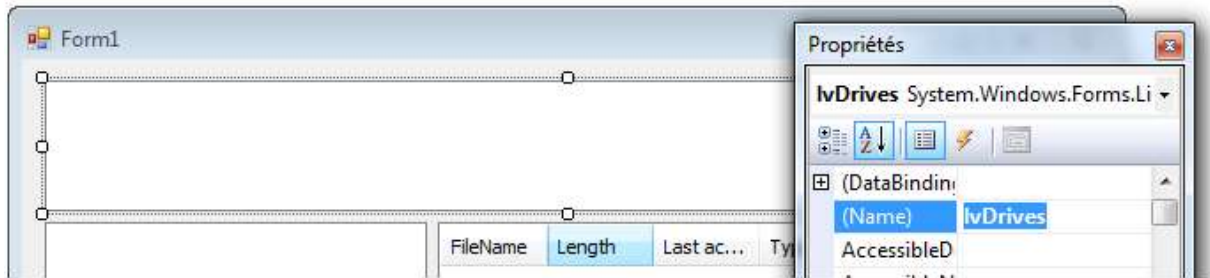
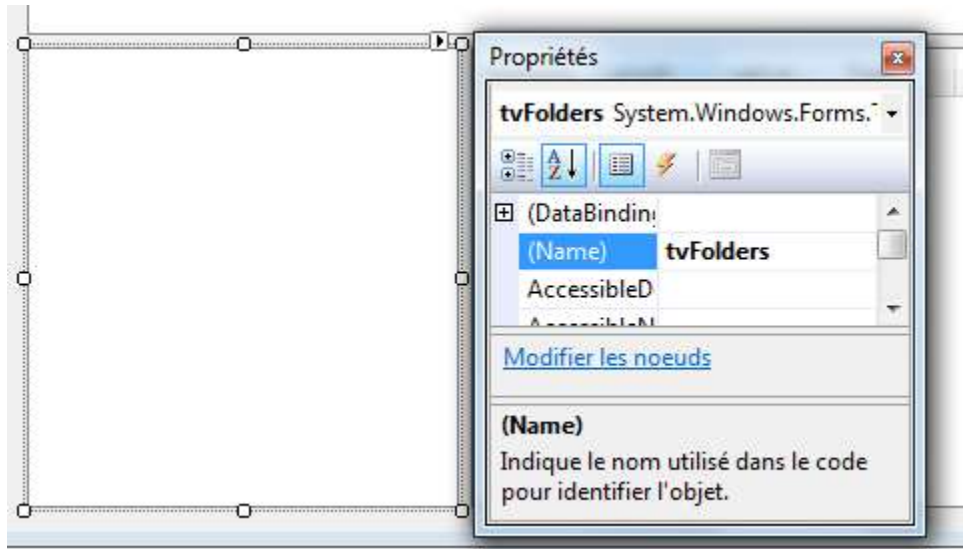
```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
        listView1.Columns.Add("Name");
        listView1.Columns.Add("Description");
        listView1.Items.Add("Folder 1", 0);
        listView1.Items.Add("Folder 2", 0);
        listView1.Items.Add("Folder 3", 0);
        listView1.Items.Add("Folder 4", 0);
        listView1.Items.Add("Folder 5", 0);
    }
}
```

قم بتنفيذ البرنامج لتشاهد النتيجة:

أتمنى أن تكون قد نجحت في إنجاز هذه الشروحات، إن لقيت أدنى صعوبة فعد إلى بداية شرح الأداة وركز جيدا لأن الأمر بسيط جدا ويتطلب فقط شيئا من الصبر.

الآن سننجز تطبيقا عمليا بهذه الأداة، إذا تمكنت من استيعابه ستكون قد قطعت شوطا مهما في برمجة الواجهات، سنقوم إن شاء الله بإنجاز متصفح الويندوز وفي غضون ذلك سنقحم مجموعة من المفاهيم السابقة مثل الإجراءات Methodes ليكون البرنامج أكثر احترافية.

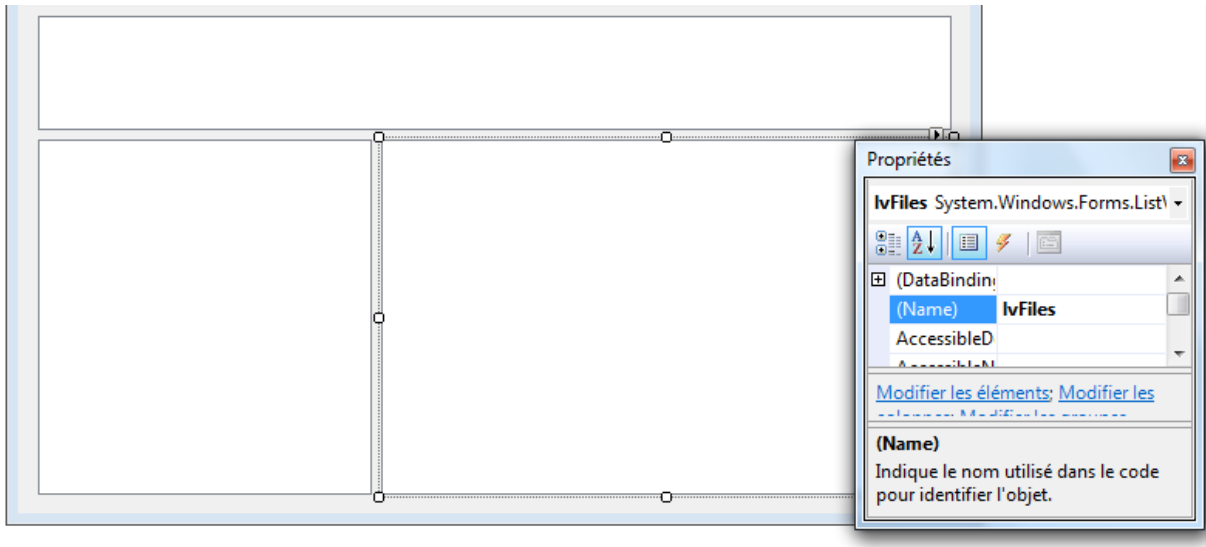
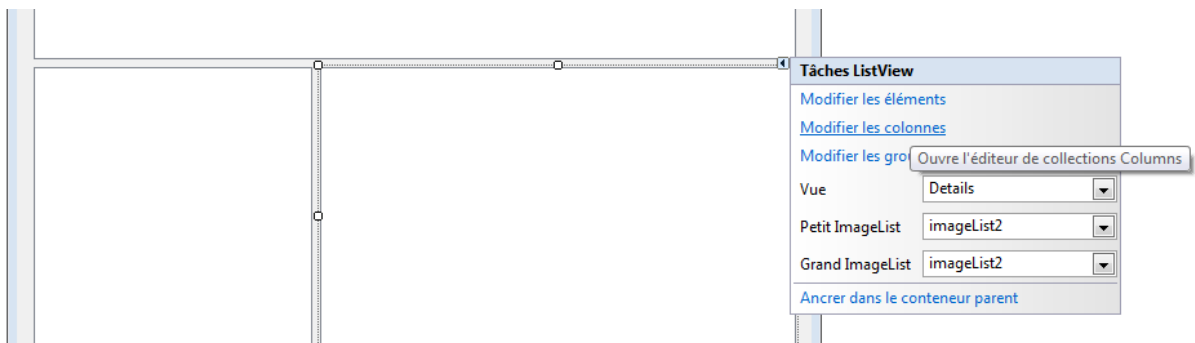
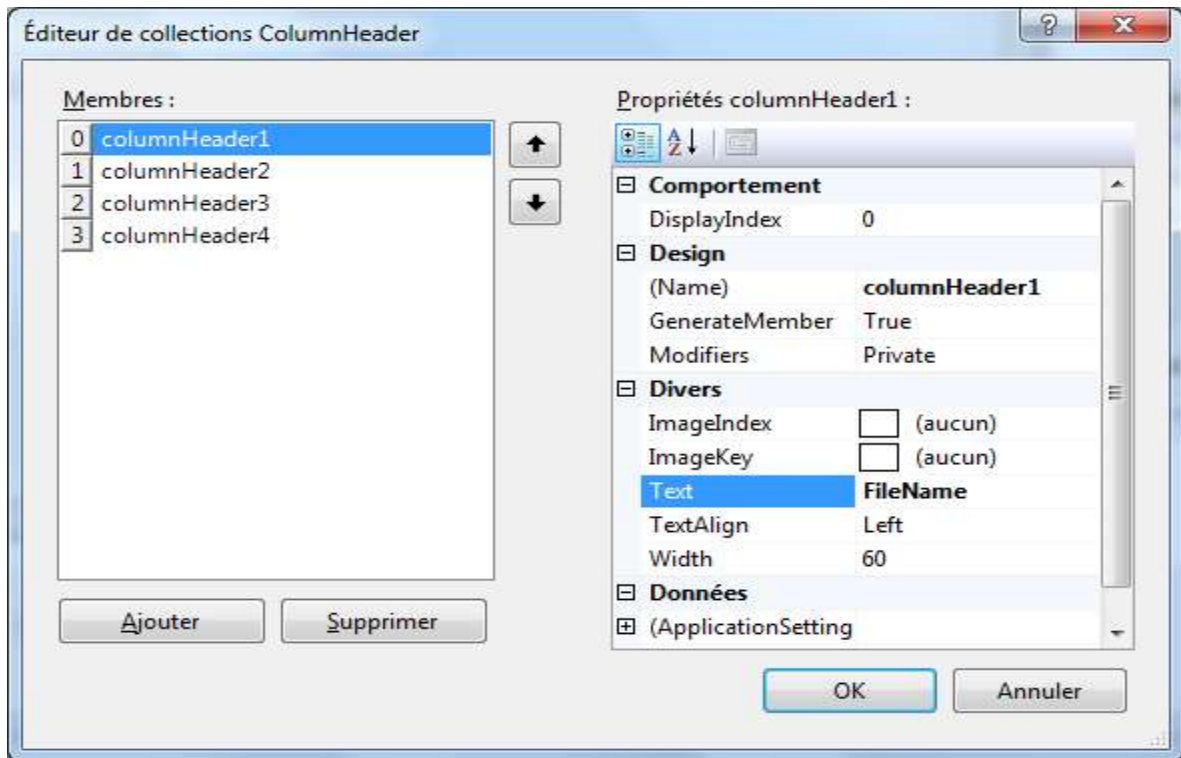
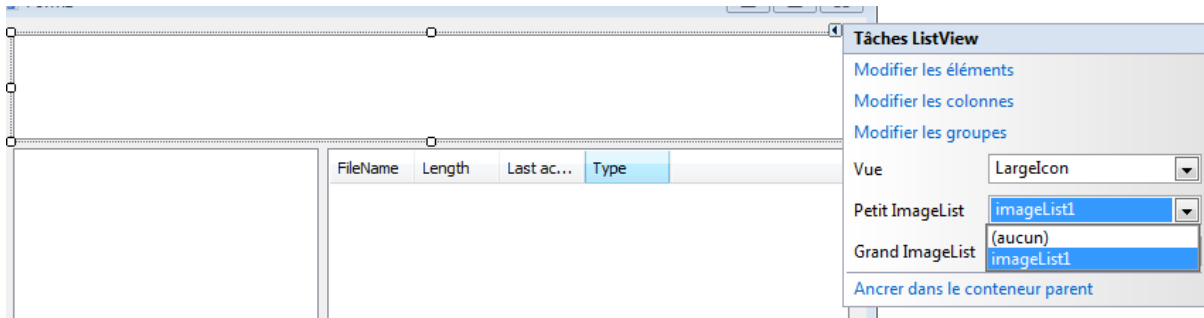
هذه صورة للبرنامج الذي سننشئه:

الأدوات التي سنحتاجها:

- قائمة العرض ListView: لعرض وحدات الحاسوب.
- شجرة العرض TreeView: لعرض مجلدات الوحدة التي يحددها المستخدم
- قائمة العرض ListView : لعرض الملفات التي يحتويها المجلد الذي يحدده المستخدم
- قائمة الصور ImageList: لتزويد الأدوات بالصور

قم أولا بإضافة ثلاث صور إلى أداة ImageList، ولتكن هذه الصور للوحدات والمجلدات والملفات.

بعد أن تضيف الصور إلى أداة ImageList1، قم بالذهاب إلى أداة ListView1 الأولى وغير اسمها Name إلى lvDrives :

بنفس الطريقة قم بتغيير اسم قائمة الشجرة إلى tvFolders:

ثم اذهب إلى قائمة العرض الثانية المخصصة لعرض الملفات، وقم بتغيير اسمها إلى lvFiles ، ثم أضف إليها الأعمدة التالية يدويا:

- اسم الملف FileName: لإظهار اسم الملف.
- حجم الملف Length: لإظهار حجم الملف.
- تاريخ آخر دخول للملف Last Access: لإظهار تاريخ آخر عملية ولوج للملف.
- نوع الملف Type: لإظهار امتداد الملف.

اتبع الصور التالية إذا استصعبت هذا الأمر:

- تغيير اسم القائمة.

- إضافة الأعمدة:

- طريقة إضافة الأعمدة وتغيير خصائصها:

الآن انتهينا من إضافة الأعمدة، سنقوم الآن بالذهاب إلى كل أداة وتحديد ImageList1 كمصدر للصور:

جيد، بعد أن تنهي التصميم خذ لك نفسا عميقا وصل على رسول الله.

" اللهم صل على محمد وعلى آل محمد كما صليت على إبراهيم وعلى آل إبراهيم، وبارك على محمد وعلى آل محمد كما باركت على إبراهيم وعلى آل إبراهيم، في العالمين إنك حميد مجيد ".

الآن سننتقل إلى كتابة الشفرة، أريد فقط أن أخبرك بأننا سنجلب مجال الأسماء System.IO الخاص بالتعامل مع الملفات والمجلدات، وهذا هو أول سطر :

```
using System;
using System.Windows.Forms;
using System.IO;
```

بعد ذلك، نقوم بالإعلان عن متغير اسمه strDrive لحفظ قيمة مسار الوحدة التي يختارها المستخدم:

```
using System;
using System.Windows.Forms;
using System.IO;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        string strDrive;
```

الآن ركز معي جيدا، سنقوم بإنشاء ثلاث إجراءات Methods:

getDrives():من خلال هذا الإجراء سنظهر كل وحدات الحاسوب في قائمة العرض lvDrives

getFolders():هذا الإجراء يقوم بعرض محتوى الوحدة التي يختارها المستخدم.

getFiles(string strPath): هذا الإجراء يقوم بعرض محتوى المجلد المحدد من طرف المستخدم.

وهذا هو نص الإجراء الأول ()getDrives:

```
void getDrives()
{
    foreach (string drive in Directory.GetLogicalDrives())
    {
        this.lvDrives.Items.Add(drive,0);
    }
}
```

قمنا بالبحث عن كل وحدات الحاسوب عن طريق الدالة ()GetLogicalDrives، التابعة للفئة Directory، ثم أضفنا الوحدات إلى القائمة lvDrives

والآن نص الإجراء الثاني ()getFolders:

```
void getFolders()
{
    strDrive = lvDrives.FocusedItem.Text;
    DirectoryInfo Dir = new DirectoryInfo(strDrive);
    tvFolders.Nodes.Clear();
    foreach (DirectoryInfo Folder in Dir.GetDirectories())
    {
        tvFolders.Nodes.Add("", Folder.Name, 1, 1);
    }
}
```

```
strDrive = lvDrives.FocusedItem.Text;
```

السطر الأول يقوم بحفظ قيمة مسار الوحدة -المحددة من طرف المستخدم- في المتغير النصي strDrive الذي أعلنا عنه قبل قليل.

```
DirectoryInfo Dir = new DirectoryInfo(strDrive);
```

في السطر الثاني قمنا بالإعلان عن متغير Dir يقوم باستخراج كل تفاصيل المسار strDrive بواسطة دوال الفئة DirectoryInfo.

```
tvFolders.Nodes.Clear();
```

في السطر الثالث نقوم بتفريغ محتوى أداة عرض الملفات tvFolders، لنملأها من جديد، إذا ألغينا هذا السطر سوف تتراكم الملفات عند كل اختيار للوحدات Drives.

```
foreach (DirectoryInfo Folder in Dir.GetDirectories())
{
    tvFolders.Nodes.Add("", Folder.Name, 1, 1);
}
```

ثم بعد ذلك نقوم بملأ القائمة الشجرية tvFolders ، بالملفات الموجودة داخل الوحدة المحددة بواسطة الدالة GetDirectories التابعة للفئة DirectoryInfo.

إذا تمكنت من استيعاب هذين الإجراءين جيدا، فتعال لنشاهد نص الإجراء الثالث، وإن كنت ترى الأمر ملتبسا فعد إلى أول التطبيق وركز جيدا، ثق بي سأنتظرك 😎

نص الإجراء الثالث getFiles(string strPath):

```
void getFiles(string strPath)
{
    ListViewItem lvi;
    DirectoryInfo Dir = new DirectoryInfo(strDrive+ strPath);
    lvFiles.Items.Clear();
    foreach (FileInfo file in Dir.GetFiles())
    {
        lvi = lvFiles.Items.Add(file.Name,2);
        lvi.SubItems.Add(file.Length.ToString());
        lvi.SubItems.Add(file.LastAccessTime.ToString());
        lvi.SubItems.Add(file.Extension);
    }
}
```

كما ترى فهذا الإجراء يستقبل برامترا من نوع نصي اسمه strPath، الذي سيعوض باسم المجلد المحدد من قبل المستخدم.

```
ListViewItem lvi;
```

أعلنا عن متغير lvi من نوع ListViewItem لكي نضع فيه ملفات المجلد المحدد، ومن ثم نظهرها في القائمة lvFiles

```
DirectoryInfo Dir = new DirectoryInfo(strDrive+ strPath);
```

في هذا السطر نقوم بجلب كل تفاصيل المجلد المحدد، وذلك بدمج مسار الوحدة Drives مع اسم المجلد.

```
lvFiles.Items.Clear();
```

هذ السطر يقوم بتفريغ قائمة الملفات عند كل نداء لهذا الإجراء تفاديا لتراكم ظهور الملفات.

```
foreach (FileInfo file in Dir.GetFiles())
{
    lvi = lvFiles.Items.Add(file.Name,2);
    lvi.SubItems.Add(file.Length.ToString());
    lvi.SubItems.Add(file.LastAccessTime.ToString());
    lvi.SubItems.Add(file.Extension);
}
```

هذا التكرار يقوم بملأ القائمة lvFiles، بمحتويات المجلد المحدد من طرف المستخدم بالاعتماد على الدالة ()GetFiles المنتمية للفئة FileInfo، وكذلك بواسطة المتغير lvi الذي يقوم بإضافة كل عنصر جديد إلى القائمة، ويقوم كذلك بترتيب البيانات حسب الأعمدة.

الآن أنهينا كتابة الإجراءات وهي المرحلة المهمة، الآن بقي فقط أن ننادي على كل إجراء في مكانه الخاص.

بالنسبة للإجراء ()getDrives، فينبغي أن ينفذ مع انطلاق البرنامج، كي تظهر الوحدات للمستخدم.

إذن فالمكان الأفضل هو بعد الدالة ()InitializeComponent:

```
public Form1()
{
    InitializeComponent();
    getDrives();
}
```

أما الإجراء الثاني فالمكان المناسب له هو الحدث Double_Click لقائمة الوحدات:

```
        private void lvDrives_DoubleClick(object sender, EventArgs e)
        {
            getFolders();
        }
```

وبالنسبة للإجراء الثالث، فسوف ننادي عليه في الحدث After_Select، التابع للأداة tvFolders

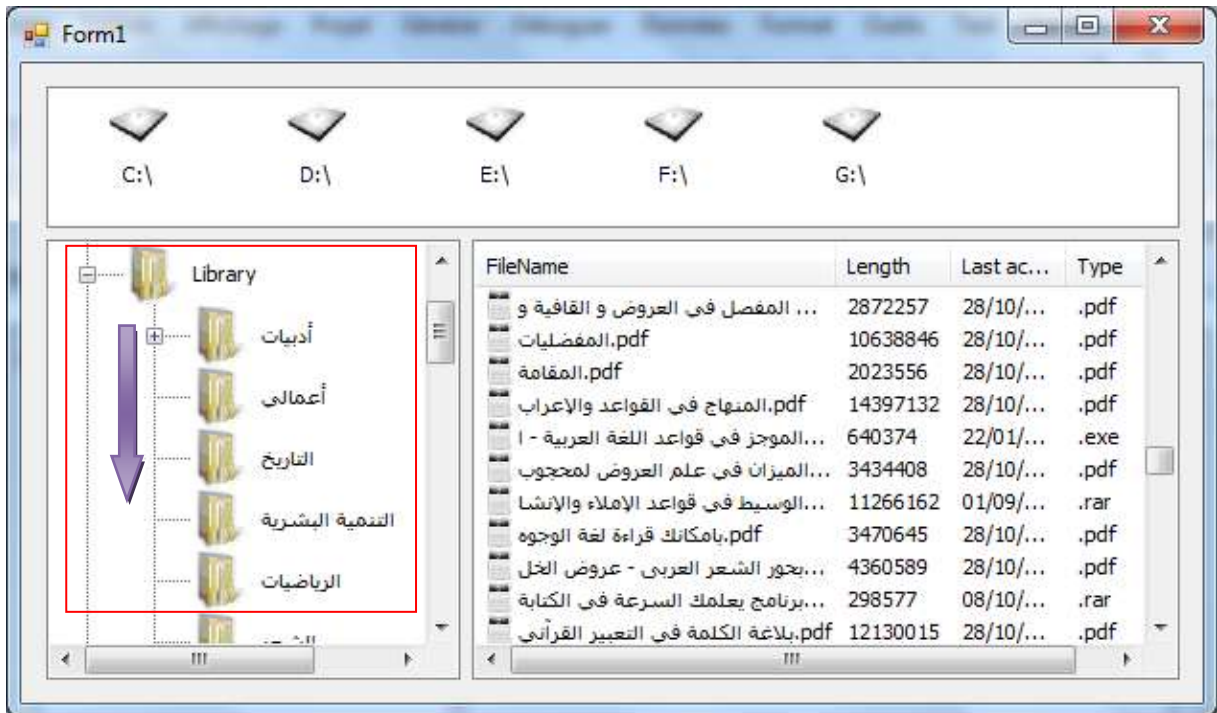
```
        private void tvFolders_AfterSelect(object sender,
TreeViewEventArgs e)
        {
            getFiles(e.Node.FullPath);

            TreeNode node;
            DirectoryInfo Dir = new DirectoryInfo(strDrive +
e.Node.FullPath);
            foreach (DirectoryInfo folder in Dir.GetDirectories())
            {
                node = new TreeNode(folder.Name, 1, 1);
                e.Node.Nodes.Add(node);
            }
        }
```

أوه !!! ماهذا الذي وضعته بعد النداء يا خالد؟؟

لا ترتبك فالأمر بسيط

```
            getFiles(e.Node.FullPath);
```

هذا السطر للنداء على الإجراء الثالث، والقيمة e.Node.FullPath تقدم لنا اسم المجلد المحدد من طرف المستخدم.

ما يأتي بعد هذا النداء هو من أجل إظهار المجلدات الفرعية للمجلد المحدد:

وهنا شرح كيف قمنا بذلك:

```
            TreeNode node;
```

في هذا السطر قمنا بالإعلان عن متغير node من نوع عقدة شجرية، لكي نحفظ فيه كل مجلد فرعي.

```
        DirectoryInfo Dir = new DirectoryInfo(strDrive + e.Node.FullPath);
```

هذا المتغير لنتمكن من جلب كل تفاصيل المجلد المحدد من قبل المستخدم.

```
foreach (DirectoryInfo folder in Dir.GetDirectories())
{
    node = new TreeNode(folder.Name, 1, 1);
    e.Node.Nodes.Add(node);
}
```

هنا نقوم بإضافة كل مجلد فرعي على شكل عقدة متشعبة من المجلد الرئيسي.

إلى هنا ينتهي التطبيق، وكما قلت لك في أوله لو استوعبته فسوف تكون قد قطعت شوطا مهما في برمجة النوافذ، لأنه على بساطته يضم تقنيات ومفاهيم مهمة كالإجراءات، ومجال الأسماء، والدوال والفئات الخاصة بالتعامل مع الوحدات والمجلدات والملفات.

سنورد الآن الشفرة كاملة للبرنامج من باب التبسيط، لأن هناك من الإخوة من قد يجد غموضا في بعض فقرات الشرح، وهذا طبيعي جدا لهذا سنعرض الشفرة كاملة، ولكن احذر من نسخها ولصقها في برنامجك لأن ذلك لي ينفعك في شيء، بل الأهم أن تكتبها بأصابعك حتى وإن لم تفهم بعض أجزائها:

```
using System;
using System.Windows.Forms;
using System.IO;
namespace WindowsExplorer
{
    public partial class Form1 : Form
    {
        string strDrive;

        //First method
        void getDrives()
        {
            foreach (string drive in Directory.GetLogicalDrives())
            {
                this.lvDrives.Items.Add(drive,0);
            }
        }

        //Second Method
        void getFolders()
        {
            strDrive = lvDrives.FocusedItem.Text;
            DirectoryInfo Dir = new DirectoryInfo(strDrive);
            tvFolders.Nodes.Clear();
            foreach (DirectoryInfo Folder in Dir.GetDirectories())
            {
                tvFolders.Nodes.Add("", Folder.Name, 1, 1);
            }
        }

        //Third Method
        void getFiles(string strPath)
        {
            ListViewItem lvi;
            DirectoryInfo Dir = new DirectoryInfo(strDrive+ strPath);
            lvFiles.Items.Clear();
            foreach (FileInfo file in Dir.GetFiles())
            {
                lvi = lvFiles.Items.Add(file.Name,2);
                lvi.SubItems.Add(file.Length.ToString());
                lvi.SubItems.Add(file.LastAccessTime.ToString());
                lvi.SubItems.Add(file.Extension);
            }
        }
```

```
        public Form1()
        {
            InitializeComponent();
            getDrives();
        }

        private void lvDrives_DoubleClick(object sender, EventArgs e)
        {
            getFolders();
        }

        private void tvFolders_AfterSelect(object sender,
TreeViewEventArgs e)
        {
            getFiles(e.Node.FullPath);
            //
            TreeNode node;
            DirectoryInfo Dir = new DirectoryInfo(strDrive +
e.Node.FullPath);
            foreach (DirectoryInfo folder in Dir.GetDirectories())
            {
                node = new TreeNode(folder.Name, 1, 1);
                e.Node.Nodes.Add(node);
            }
        }

    }
}
```

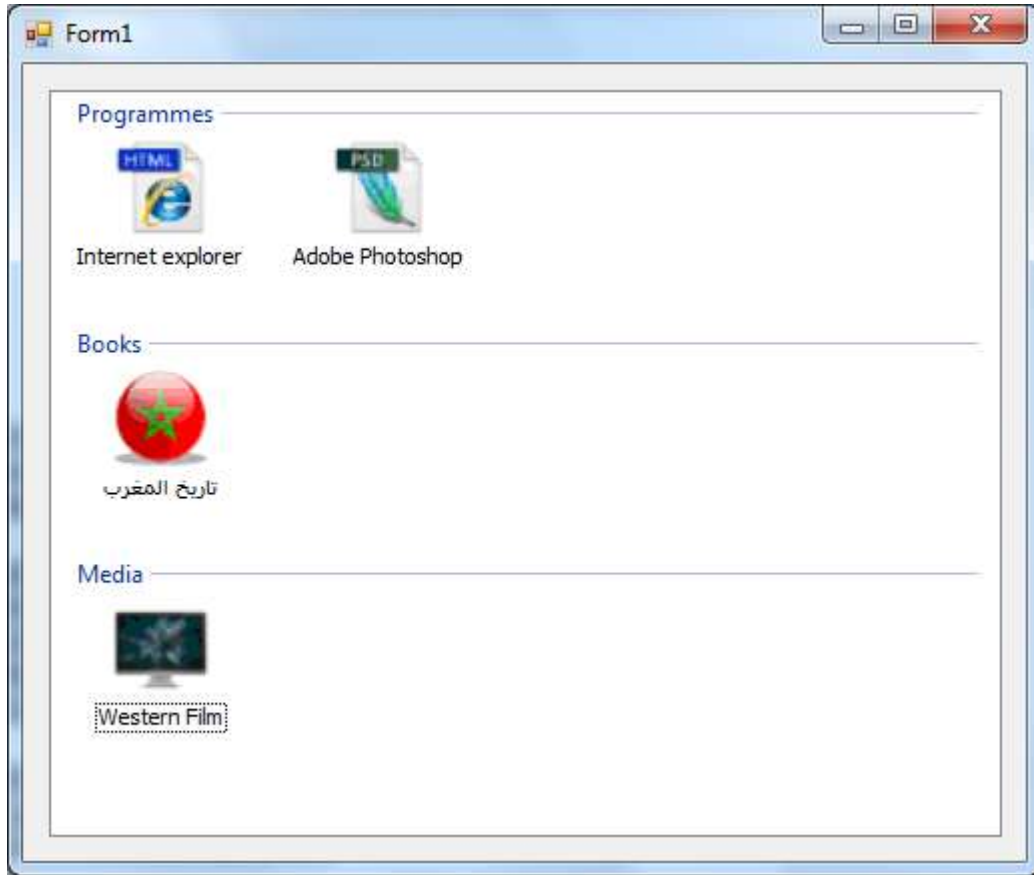
بقي لنا شيء واحد وننتهي من موضوع الأداة ListView، وهو فرز الملفات على شكل مجموعات Grouping، كما يبدو جليا في الصورة التالية:

Nom	Modifié le	Type	Taille
E – K (1)			
Islamic Multimedia	10/12/2012 00:55	Dossier de fichiers	
L – P (4)			
Library	10/12/2012 00:56	Dossier de fichiers	
Music & Videos	10/12/2012 14:26	Dossier de fichiers	
Old Storage	10/12/2012 00:48	Dossier de fichiers	
Programming Refrences (Books)	10/12/2012 14:47	Dossier de fichiers	
Q – Z (1)			
References (Multimedia)	10/12/2012 14:42	Dossier de fichiers	

الأمر سهل جدا، كل ما علينا هو أن ننشىء المجموعة، ثم نقوم بضم عناصرها إليها، فلنقم بذلك على عجل، أنشىء مشروع جديد، وأضف للفورم أداة ListView وأداة ImageList وقم بتزويدها بالصور التي تريد إظهارها، ثم اربط الأداتين مع بعض كما رأينا سابقا.

الآن أدخل إلى نافذة الكود واكتب ما يلي بعد الدالة InitializeComponent():

```
//Create 3 groups
listView1.Groups.Add("", "Programmes");
listView1.Groups.Add("", "Books");
listView1.Groups.Add("", "Media");

//Add 2 items to the first group
listView1.Items.Add("Internet explorer", 1);
listView1.Items[0].Group = listView1.Groups[0];
listView1.Items.Add("Adobe Photoshop", 0);
listView1.Items[1].Group = listView1.Groups[0];

//Add 1 item to the second group
listView1.Items.Add("تاريخ المغرب", 3);
listView1.Items[2].Group = listView1.Groups[1];


//Add 1 item to the third group
listView1.Items.Add("Western Film", 2);
listView1.Items[3].Group = listView1.Groups[2];
```

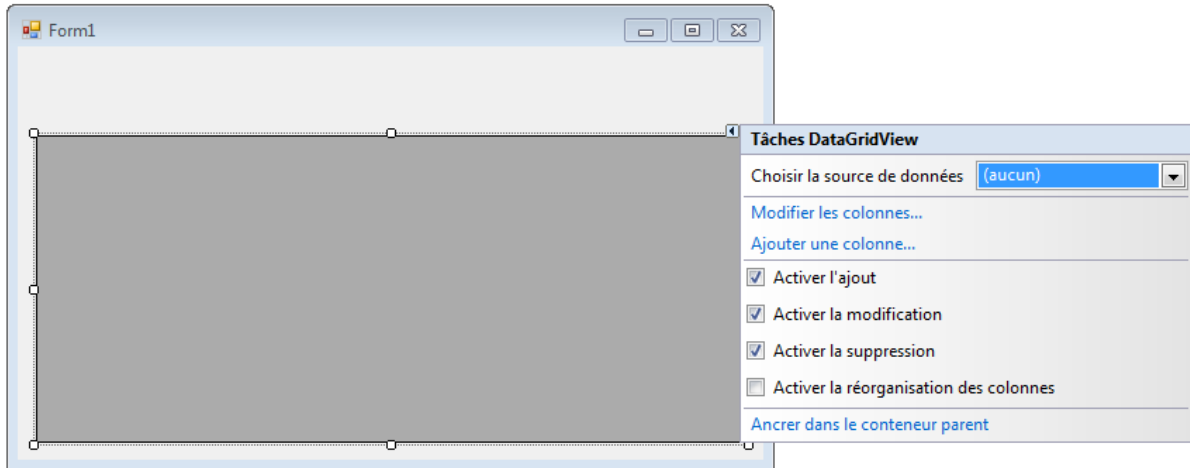
قمنا في الأول بإنشاء ثلاث مجموعات، ثم أضفنا لكل مجموعة عناصرها مع تحديد النص والصورة المراد ظهورها، إذا طبقت ذلك كما ينبغي ستحصل على نتيجة كهذه:

أتمنى أن تكون الأمور جد واضحة.

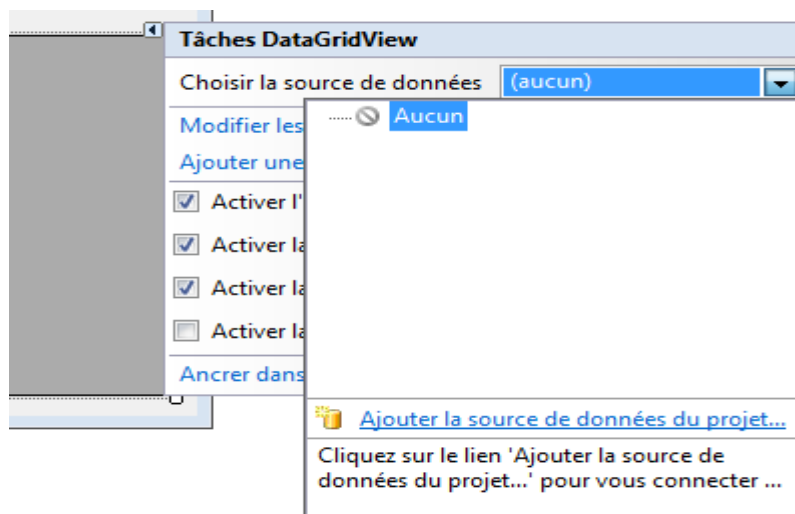
12. قائمة عرض البيانات DataGridView

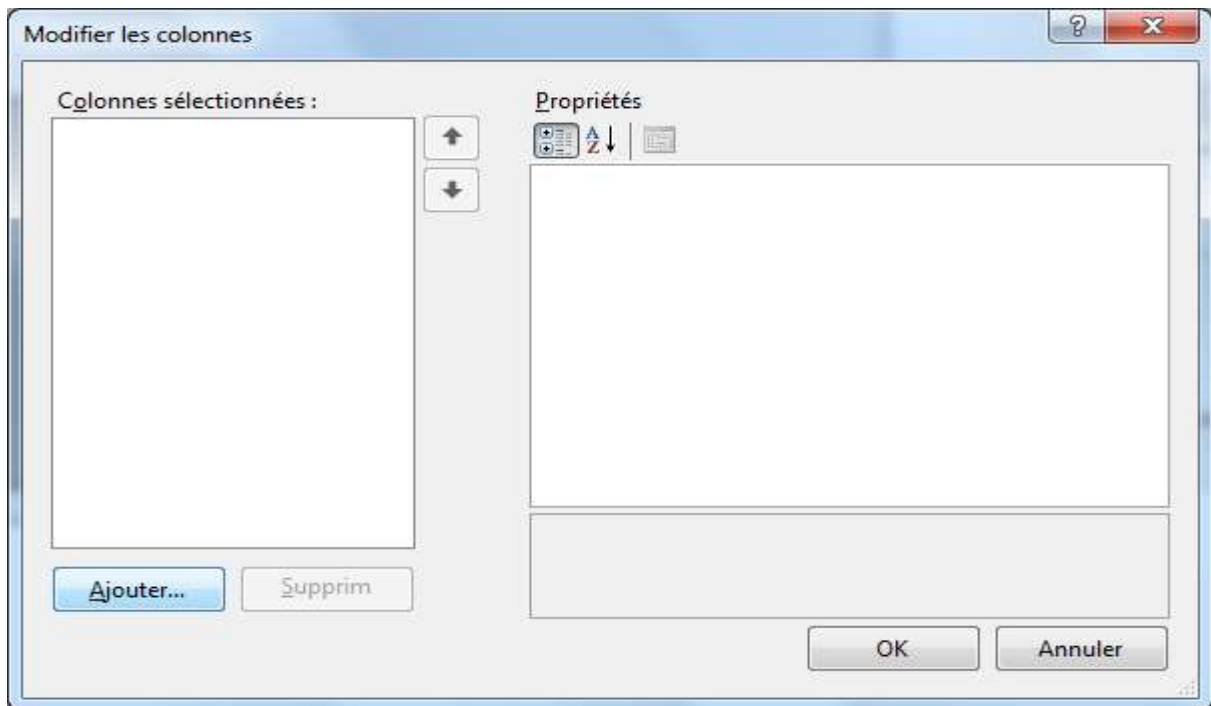
تستعمل هذه الأداة غالبا لعرض معلومات قادمة من قاعدة بيانات، وهي شبيهة ببرنامج ميكروسوفت إكسيل Excel، حيث تحتوي على خلايا وأعمدة.

الجزء المهم من هذه الأداة سنتعرف عليه في إبانه، حينما نلج إلى قواعد البيانات، أما الآن فسوف نكتفي بالتعرف إليها وفهم طريقة الاشتغال بها، وإنجاز بعض التطبيقات لكي نستأنس بها.

وهذه صورة لأداة DataGridView، وهي معبأة من جدول بقاعدة بيانات:

تتوفر هذه الأداة على عدد كبير من الخصائص، التي تيسر للمستخدم مهامه، وتجعل البرنامج أكثر احترافية، طيب لنقم أولا بإنشاء مشروع جديد ونضيف إليه أداة DataGridView.

عند الضغط على السهم الصغير الموجود أعلى يمين الأداة ستظهر لك هذه النافذة الصغيرة، وهذه شروح لأهم ما فيها:

اختيار مصدر البيانات Choose Data Source: هذا الخيار لتحديد مصدر البيانات يدويا، إذا ضغطت عليه سوف تظهر لك هذه النافذة:

بعد ذلك تستطيع اختيار مصدر البيانات.

- تعديل الأعمدة Edit Columns: هذا الخيار إذا أردت تعديل الأعمدة، تماما كما رأينا مع ListView، إذا ضغطت عليه ستظهر لك النافذة التالية، التي من خلالها تستطيع إضافة وتعديل ما تشاء من الأعمدة:

حينما تضغط على الزر Add، ستظهر لك هذه النافذة، التي من خلالها تستطيع إعطاء اسم العمود، ونوعه وقيمته النصية:

Ajouter une colonne

Colonne liée aux données

Colonnes du DataSource

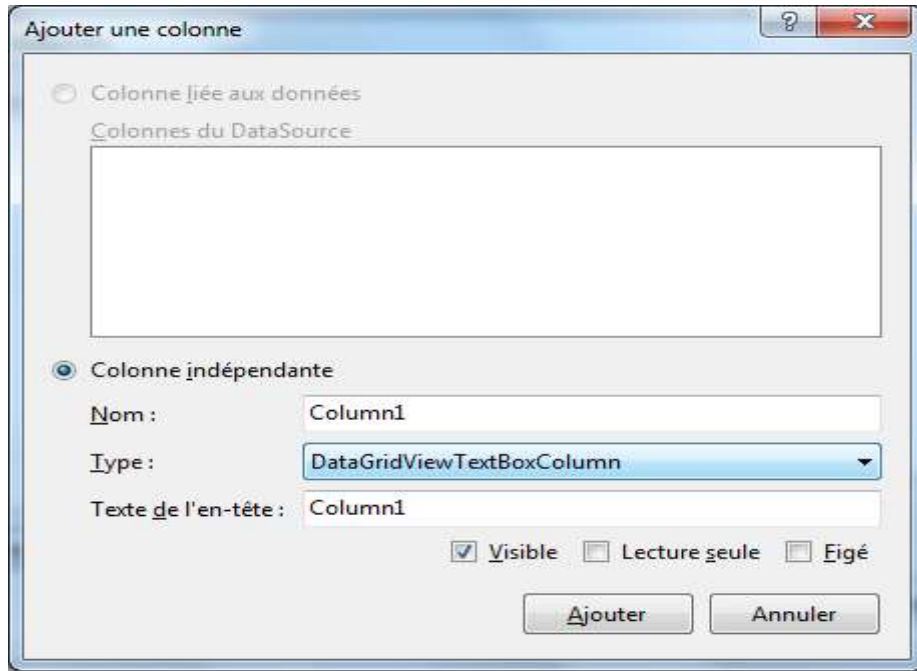
Colonne indépendante

Nom : Column1

Type : DataGridViewTextBoxColumn

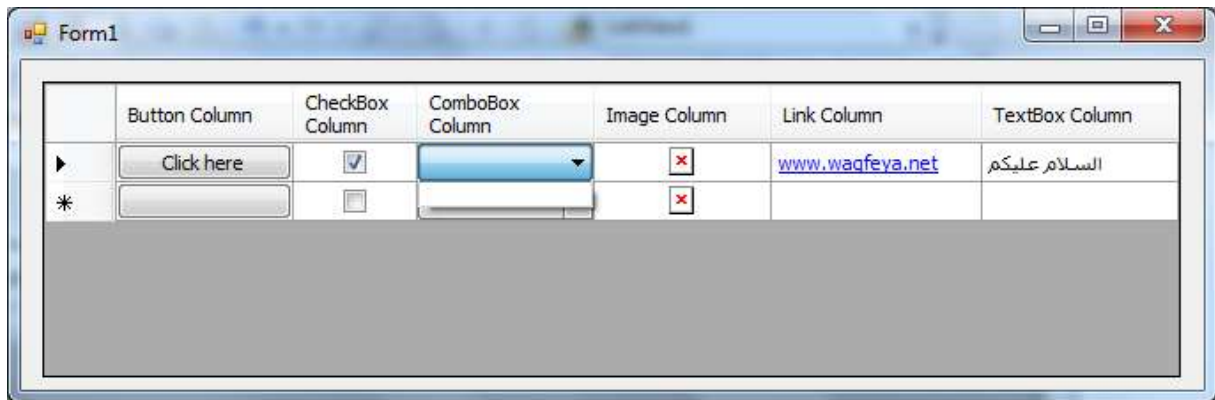
Texte de l'en-tête : Column1

Visible   Lecture seule   Figé

Ajouter   Annuler

بالنسبة لأنواع العمود فهي كما يلي:

- DataGridViewButtonColumn: لإظهار محتوى العمود على شكل زر Button
- DataGridViewCheckBoxColumn: لإظهار محتوى العمود على شكل علبة التحديد CheckBoxe
- DataGridViewComboBoxColumn: لإظهار محتوى العمود على شكل علبة الكومبو ComboBox
- DataGridViewImageColumn:لإظهار محتوى العمود على شكل صورة Image

- DataGridViewLinkColumn:لإظهار محتوى العمود على شكل رابط ديناميكي Link
- DataGridViewTextBoxColumn: وهي الحالة الافتراضية حيث يكون محتوى العمود عبارة عن علبة نص TextBox

وهذه صورة توضيحية لمختلف أنواع الأعمدة:

- إضافة الأعمدة Add Columns: لإضافة الأعمدة.
- تفعيل إضافة البيانات Enable Adding: لتفعيل الإضافة أو إلغائها، لو قمت بإلغاء التحديد، فلن تستطيع إضافة سطر جديد.
- تفعيل تعديل البيانات Enable Editing: لتفعيل أو إلغاء تعديل محتوى خانات الأداة.

- تفعيل حذف البيانات Enable Deleting: لتفعيل أو إلغاء حذف الأسطر، لو قمت بإلغاء هذه الخاصية فلن يستطيع المستخدم حذف سطر من أداة DataGridView
- تفعيل إعادة ترتيب الأعمدة Enable Column Reordering: وتكون افتراضيا ملغية، لو قمت بتحديدها، فستتيح للمستخدم إمكانية إعادة ترتيب أعمدة DataGridView

تعبئة DataGridView بواسطة DataTable

الآن سننجز تطبيقا سهلا يقوم بتعبئة أداة DataGridView بمصدر بيانات من نوع DataTable، وهذه فرصة مناسبة لكي نتعرف على هذه الفئة لأننا سنراها باستمرار في الجزء الخاص بقواعد البيانات إن شاء الله، وهذه صورة للنتيجة المتوخاة من التطبيق:

Name	Address	Birth Date
Khalid ESSAADANI	Morocco	18/05/1989 00:00:00
Hamid MAKBOUL	France	22/10/1989 00:00:00
Mohamed ELKHAL	Canada	01/09/1988 00:00:00
Younes MAADANE	Italia	12/07/0187 00:00:00

تعريف فئة جدول البيانات DataTable:

شبيهة إلى حد بعيد بمفهوم المصفوفات Arrays، لأنها تتوفر أيضا على أعمدة Columns وأسطر Rows، ولكنها أكثر تفاعلا مع البيانات من المصفوفات، كما أن لها مجموعة من الخصائص والمزايا التي تجعل المبرمجين يميلون إليها ويفضلونها على المصفوفات عند تعاملهم مع البيانات.

وتقع هذه الفئة تحت مجال الأسماء System.Data، الذي يكون مجلوبا افتراضيا، إن لم تجده مجلوبا فقم بجلبه.

إذن باختصار فهذه الفئة عبارة عن جدول ثنائي الأبعاد يخصص للتعامل مع البيانات إما المحفوظة في قواعد البيانات، أو المحفوظة في الذكرة الحية.

سنقوم أولا بالإعلان عن كائن جديد من هذه الفئة ولنسمه dTable:

```
DataTable dTable = new DataTable();
```

بعد ذلك سنشرع في إضافة الأعمدة DataColumn إلى الكائن dTable:

```
DataColumn Name = new DataColumn("Name",typeof(string));
dTable.Columns.Add(Name);

DataColumn Address = new DataColumn("Address",typeof(string));
dTable.Columns.Add(Address);

DataColumn BirthDate = new DataColumn("Birth Date",typeof(DateTime));
dTable.Columns.Add(BirthDate);
```

النوع DataColumn يمثل عمود Column يمكن إضافته إلى جدول بيانات DataTable

قمنا بإنشاء ثلاثة أعمدة، وقمنا بتحديد نوع كل عمود عن طريق الدالة ()TypeOf، وبعد أن ننشيء كل عمود نقوم بإضافته إلى dTable.

الآن قمنا بإنشاء جدول فارغ من نوع DataTable، يتكون من ثلاثة أعمدة، بقي لنا فقط أن نقوم بتعبئة هذا الجدول dTable بمجموعة من الأسطر Rows:

```
dTable.Rows.Add(new object[] { "Khalid", "Morocco", new DateTime(1989, 5, 18) });

dTable.Rows.Add(new object[] { "Hamid", "France", new DateTime(1989, 10, 22) });

dTable.Rows.Add(new object[] { "Mohamed", "Canada", new DateTime(1988, 9, 1) });
```

الخاصية Rows، تمثل أسطر جدول البيانات dTable، وتحتوي على مجموعة من الدوال التي تقوم بالإضافة والحذف والبحث...

قمنا بإضافة أربعة أسطر إلى جدول البيانات dTable، وقمنا بإعطاء قيمة كل خانة، تفاديا لإطالة الكود قمنا بجمع كل التفاصيل في مصفوفة من نوع object، ثم أعطينا كل عنصر قيمته، بإمكاننا فعل ذلك بطريقة أسهل لكنها ستكون أطول.

الآن لدينا جدول بيانات مكون من ثلاثة أعمدة، ويضم أربعة أسطر، بقي فقط أن نجعله مصدرا لأداتنا DataGridView:

```
this.dataGridView1.DataSource = dTable;
```

أتمنى أن تكون قد استوعبت التطبيق جيدا، وهذا هو نصه كاملا:

```
namespace DataGridViewTest
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            DataTable dTable = new DataTable();

            DataColumn Name = new DataColumn("Name",typeof(String));
            dTable.Columns.Add(Name);
            DataColumn Address = new
DataColumn("Address",typeof(String));
            dTable.Columns.Add(Address);
            DataColumn BirthDate = new DataColumn("Birth
Date",typeof(DateTime));
            dTable.Columns.Add(BirthDate);

            dTable.Rows.Add(new object[] { "Khalid ESSAADANI", "Morocco",
new DateTime(1989, 5, 18) });
            dTable.Rows.Add(new object[] { "Hamid MAKBOUL", "France", new
DateTime(1989, 10, 22) });
            dTable.Rows.Add(new object[] { "Mohamed ELKHAL", "Canada",
new DateTime(1988, 9, 1) });
            dTable.Rows.Add(new object[] { "Younes MAADANE", "Italia",
new DateTime(187, 7, 12) });

            this.dataGridView1.DataSource = dTable;
        }
    }
}
```

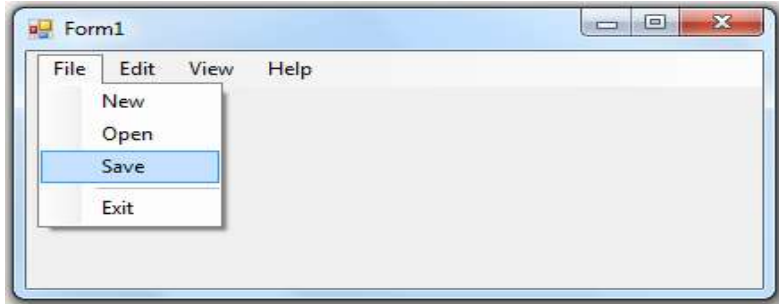
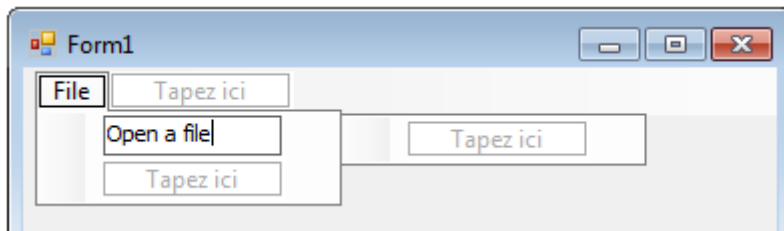
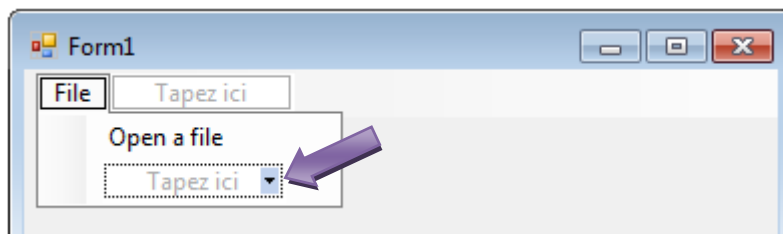
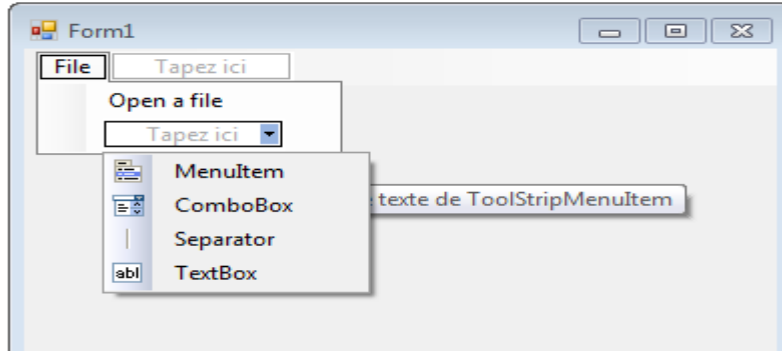
سنقوم الآن بعرض بعض خصائص أداة DataGridView:

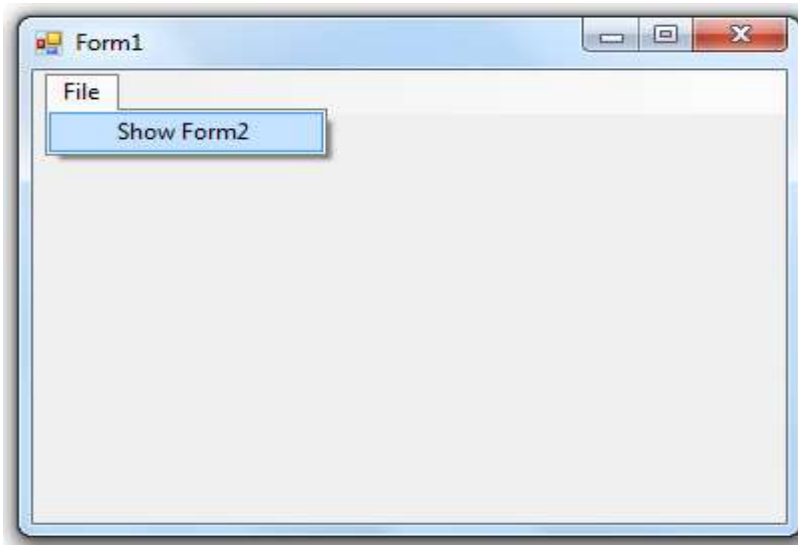
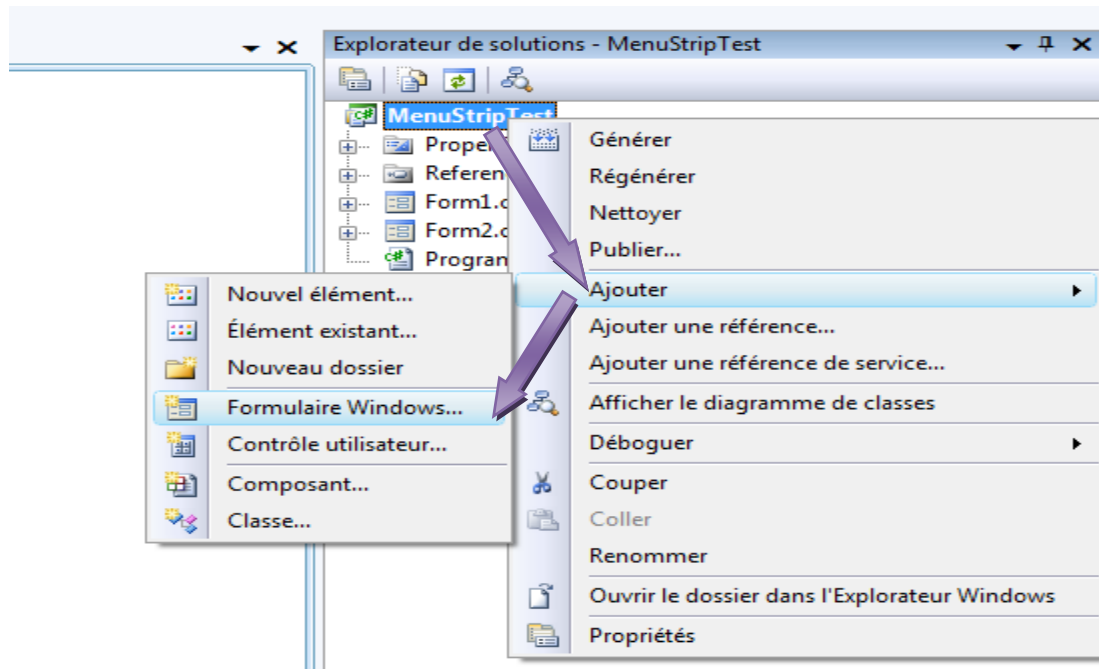
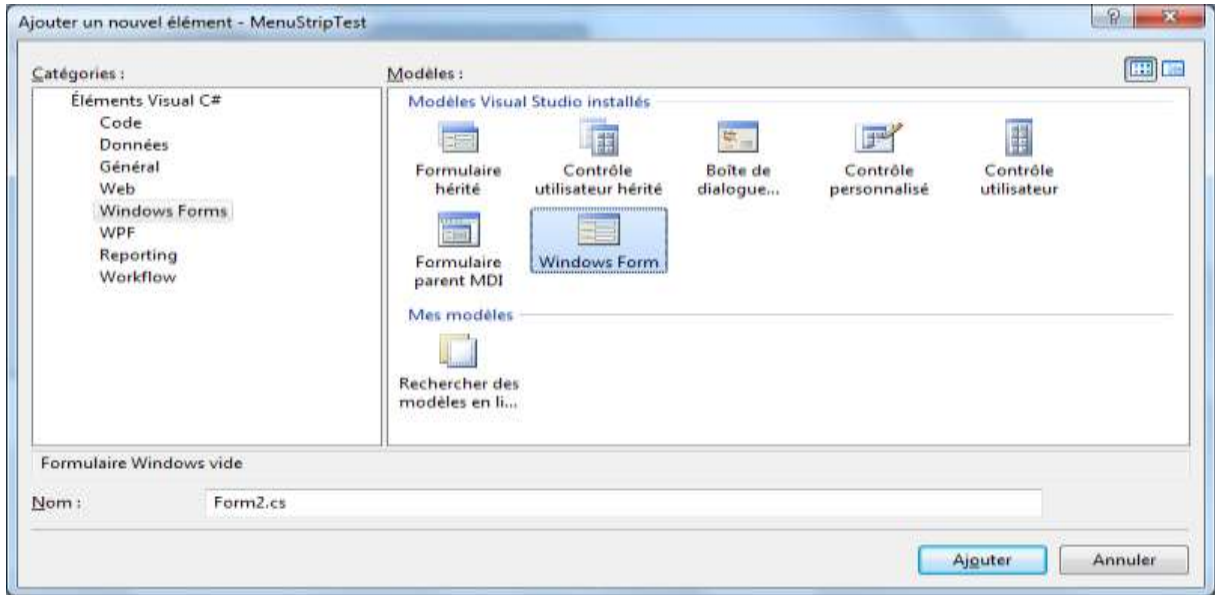
الخصيصة	دورها
AllowUserToAddRows	لتمكين المستخدم من الإضافة أو منعه وتأخذ إما True أو False
AllowUserToDeleteRows	لتمكين المستخدم من الحذف أو منعه وتأخذ إما True أو False
AllowUserToOrderColumns	لتمكين المستخدم من ترتيب الأعمدة أو منعه وتأخذ إما True أو False
AllowUserToResizeColumns	لتمكين المستخدم من تغيير مقاس الأعمدة أو منعه وتأخذ إما True أو False
AllowUserToResizeRows	لتمكين المستخدم من تغيير مقاس الأسطر أو منعه وتأخذ إما True أو False
AlternatingRowsDefaultCellStyle	لتغيير خصائص الأسطر المتتالية، كما يبدو جدول الخصائص الآن، حيث تجد لون السطر في تناوب.
AutoSizeColumnsMode	لتغيير طريقة عرض الأعمدة، فمثلا لو أردت أن تملأ الأعمدة كل DataGridView، فما عليك سوى اختيار القيمة Fill

BackGroundColor	لتغيير لون خلفية الأداة.
DefaultCellStyle	لتغيير خصائص الخانات Cells
MultiSelect	لتفعيل أو إلغاء التحديد المتعدد للأسطر.
ReadOnly	لجعل أداة DataGridView للقراءة فقط، بحيث لا يستطيع المستخدم إجراء أي تعديل أو إضافة عليها.
ScrollBars	لإظهار أشرطة التمرير سواء العمودية أو الأفقية.
SelectionMode	لتغيير طريقة تحديد السطر، فمثلا لو (اردت أن يحدد السطر بأكمله وليس الخانة فقط، قم باختيار القيمة FullRowSelect

حاول تجريب هذه الخصائص لتتعرف عليها أكثر.

13. أداة القائمة الرئيسية MenuStrip

وتعد هذه الأداة من أهم الأدوات التي يحفل بها معظم البرامج، وهذه صورة لها:

من علبة الأدوات، قم بجذب أداة القائمة MenuStrip إلى الفورم، جرب أن تضيف بعض القوائم الرئيسية، ثم أضف قوائم فرعية لها:

إذا مررت بالمؤشر فوق مكان كتابة اسم القائمة، سيظهر لك سهم صغير موجه نحو الأسفل، كما تظهر الصورة التالية:

إذا قمت بالضغط عليه ستظهر لك الاختيارات التالية:

الاختيار الأول يتيح لك إضافة قائمة من نوع MenuItem وهي النوع الافتراضي، والثاني يتيح لك إضافة قائمة من نوع كومبو ComboBox، والثالث يقوم بإضافة فاصل بين القوائم الفرعية، والأخير يتيح إضافة قائمة تسمح بالكتابة فيها.

سنقوم الآن إن شاء الله بإنجاز تطبيق يضم نافذتين Forms، وسنضع في الفورم الأول أداة MenuStrip، التي من خلالها سنقوم بفتح الفورم الثاني، كما توضح الصورة التالية:

بعد أن تضيف أداة MenuStrip إلى الفورم Form1، وبعد أن تقوم بإضافة القائمة الرئيسية File والقائمة الفرعية Show Form2،سنقوم بإضافة الفورم الثاني، اذهب إلى متصفح المشروع، واضغط بيمين الماوس على الملف الرئيسي واختر Add، ثم اختر Windows Form، كما تظهر الصورة:

أو باختصار اضغط على Ctrl+Shift+A لتطالعك النافذة التالية:

قم بإعطاء الفورم أي اسم تريد أو اتركه كما هو Form2، بعد أن تضغط زر OK، ستلاحظ أنه صار لديك 2 فورم في المشروع، قم بالعودة إلى الفورم الأول، وادخل إلى نافذة الكود الخاصة به.

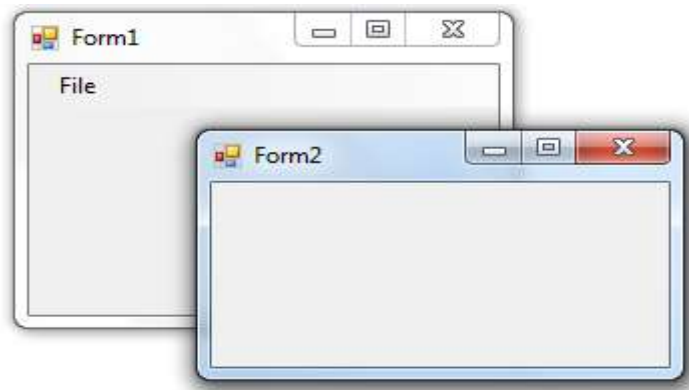
قم بالإعلان عن متغير من نوع الفورم الثاني في الحيز الخاص بالإعلان:

```
Form2 frm;
```

بعد ذلك، عد إلى الفورم، وقم بتحديد القائمة الفرعية Show Form2، ثم اضغط عليها مرتين لتنتقل إلى الحدث Click:

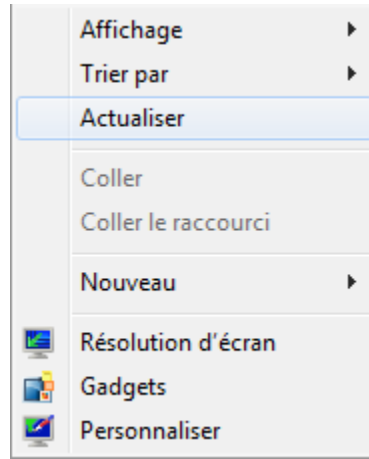
```
private void ShowForm2ToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            frm = new Form2();
            frm.Show();
        }
```

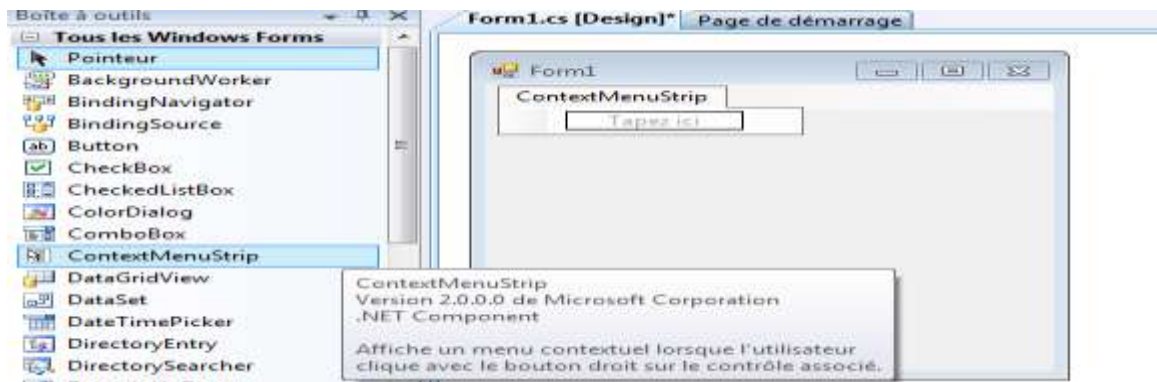
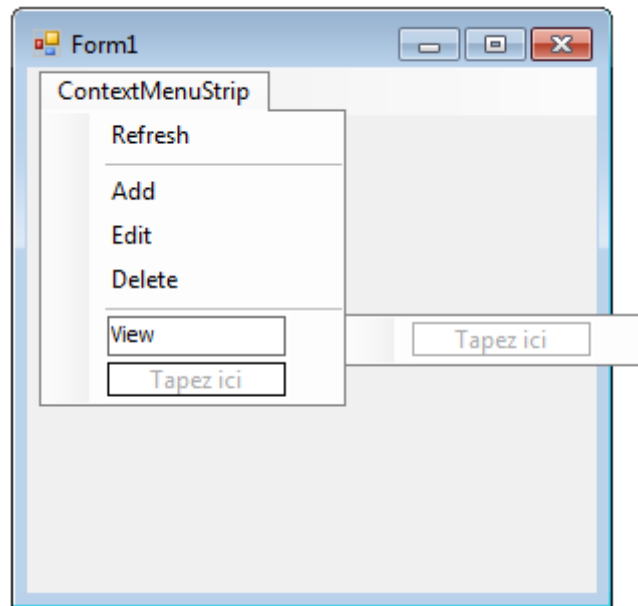
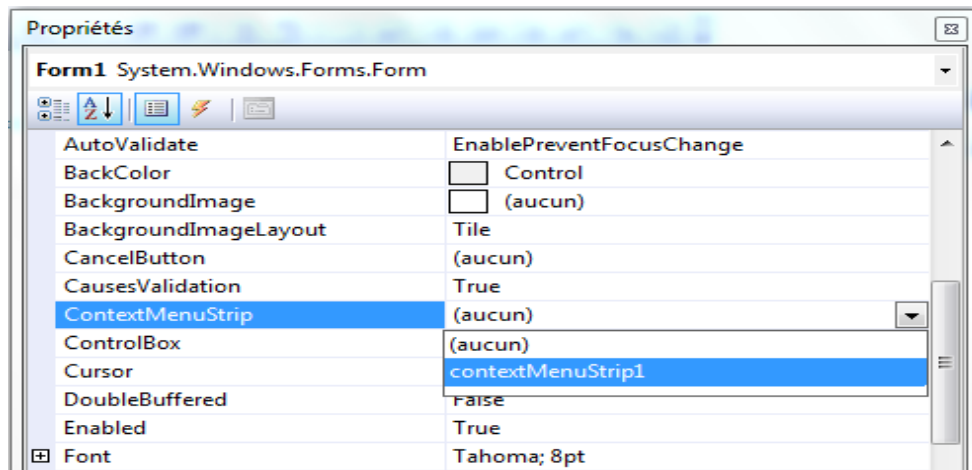
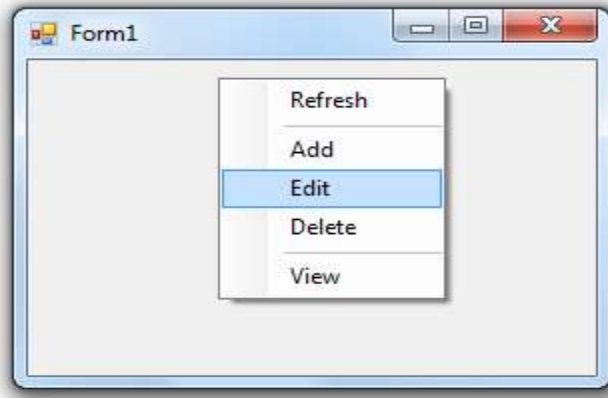
في السطر الأول قمنا بإنشاء كائن من Form2، ثم قمنا بإظهاره عن طريق الدالة Show().

إذا قمت بالتنفيذ، ستظهر لك النتيجة التالية عند ضغطك على النافذة الفرعية Show Form2:

ستلاحظ أن الفورم الثاني يظهر فوق الفورم الأول، سنرى فيما بعد كيفية جعل الفورم الثاني ابنا للفورم الأول، بحيث لا يستطيع أن يخرج عنه كما في البرامج العادية، حيث تكون النوافذ الفرعية داخل النافذة الأم لا تتجاوزها.

بإمكانك إضافة صور إلى القوائم، وبإمكانك تغيير خصائصها كما يحلو لك من نافذة الخصائص.

14. أداة القائمة المنسدلة ContextMenuStrip

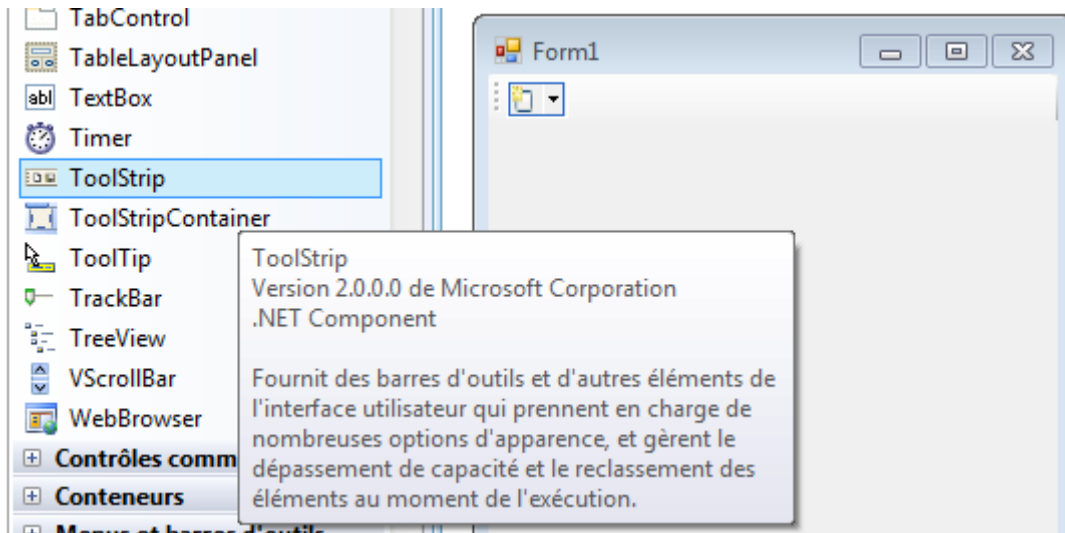
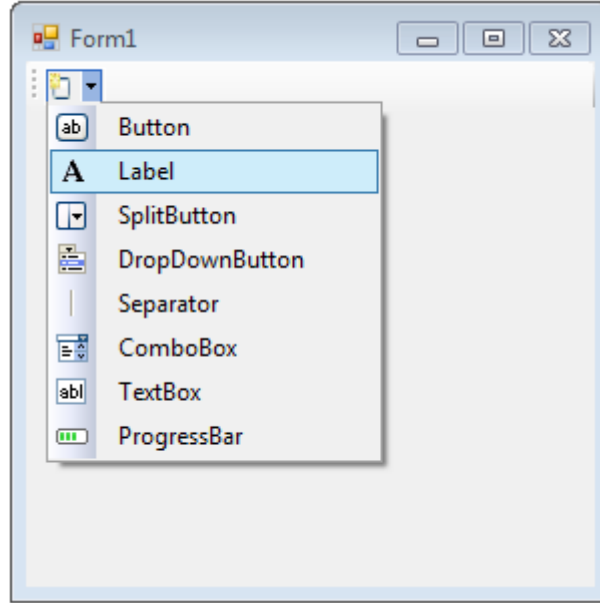
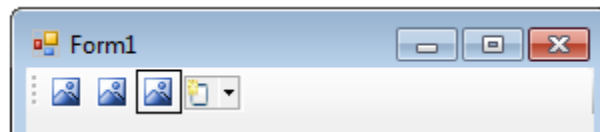
طبعا سبق لك وأن رأيت مثل هذا:

يسمى هذا النوع من القوائم بالقوائم المنسدلة ContextMenuStrip ، الذي يظهر عندما تضغط على يمين الماوس، وقلما تجد برنامجا يخلو منه.

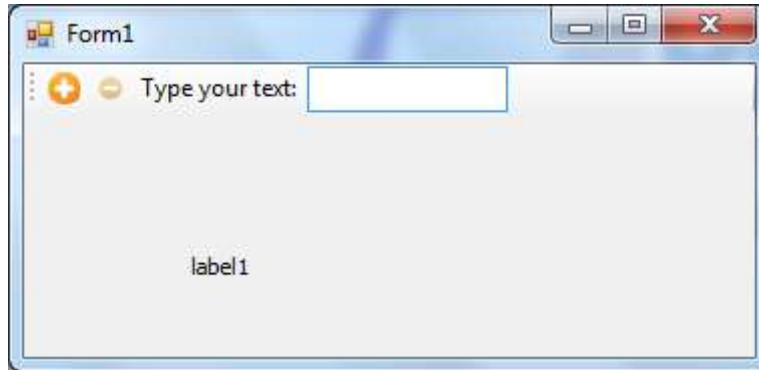
قم بإنشاء مشروع جديد ثم قم بسحب هذه الأداة على الفورم:

قم بإضافة القوائم إليها بنفس الطريقة التي رأينا مع أداة MenuStrip:

بعد أن تضيف القوائم التي تريد، قم بتحديد الفورم، واذهب إلى خصائصه، ستجد خصيصة اسمها ContextMenu، من خلالها قم بتحديد أداة ContextMenuStrip1

الغاية من ربط هذه الأداة مع الفورم، هو إظهار القوائم حينما يضغط المستخدم بيمين الماوس على الفورم، قم بالتنفيذ، واضغط بيمين الماوس على الفورم وشاهد النتيجة:

ملحوظة:

بإمكانك ربط أداة ContextMenuStrip مع معظم الأدوات لإظهار قوائم عليها عند الضغط بيمين الماوس، هذه صورة لإحدى البرامج التي قمت فيها بالربط بين هذه الأداة و أداة DataGridView:

15. أداة شريط الأدوات ToolStrip

غالبا ما يكون الغرض من هذه الأداة هو عرض محتوى القوائم الفرعية بالصور وبشكل واضح يجعل المستخدم أكثر ارتياحا، وهذه صورة الأداة:

قم بسحب أداة ToolStrip على الفورم:

اضغط على السهم الصغير الموجود على الأداة لترى النافذة التالية:

تستطيع أن تضيف أي أداة تشاء، ولكن في الغالب يفضل اختيار الأداة Button، لأنها الأنسب في حالة القوائم:

تستطيع تغيير صور الأزرار كما تشاء من خلال تحديد الزر والذهاب إلى نافذة خصائصه.

سنقوم الآن بإنجاز تطبيق بسيط لكي نتعرف أكثر على هذه الأداة، سنقوم بإنشاء مشروع جديد، ثم نضيف أداة ToolStrip، بعد ذلك نختار منها زرين 2 Buttons و علبة نص TextBox، كما تظهر الصورة التالية:

أضف أداة Label إلى الفورم.

يقوم المستخدم بكتابة نص في علبة النص التابعة للأداة ToolStrip1، فيظهر هذا النص على أداة Label1، ويمكن للمستخدم أن يزيد وينقص في حجم الخط بالاعتماد على الزرين + و -

قم بتحديد علبة النص، ثم اذهب إلى نافذة الأحداث واختر الحدث TextChanged، واضغط عليه مرتين، لتنتقل إلى نافذة الكود المتعلقة به ، واكتب فيه هذا السطر:

```
private void toolStripTextBox1_TextChanged(object sender, EventArgs e)
        {
            this.label1.Text = toolStripTextBox1.Text;
        }
```

تكمن أهمية هذا الحدث TextChanged في كونه ينفذ عند الكتابة في علبة النص، وبالتالي فكل ما ستكتبه فيها سيظهر في نفس الوقت على Label1.

ملحوظة:

بالنسبة للاسم toolStripTextBox1 فهو يعطى افتراضيا لعلبة النص التابعة لأداة ToolStrip ويمكنك تغييره إذا أحببت عن طريق نافذة الخصائص.

جيد، إذا نفذت البرنامج الآن، وكتبت في علبة النص ستلاحظ أن كتابتك تظهر في نفس الوقت على أداة Label1:

قم أولا بالإعلان عن متغير من نوع رقمي اسمه size، أعلن عنه هنا:

```
public partial class Form1 : Form
{
    int size = 12;
    public Form1()
    {
        InitializeComponent();
    }
```

الغاية من هذا المتغير الرقمي هو الزيادة والنقصان في حجم الخط، وأعطيناه القيمة البدئية 8 لأنها القيمة الافتراضية لحجم الخط.

الآن اذهب إلى زر زيادة حجم الخط، واضغط عليه مرتين للولوج إلى الحدث Click الخاص به، أو عن طريق نافذة الأحداث، ثم اكتب ما يلي:

```
private void toolStripButton1_Click(object sender, EventArgs e)
{
    this.label1.Font =new Font("Tahoma", size);
    size += 4;
}
```

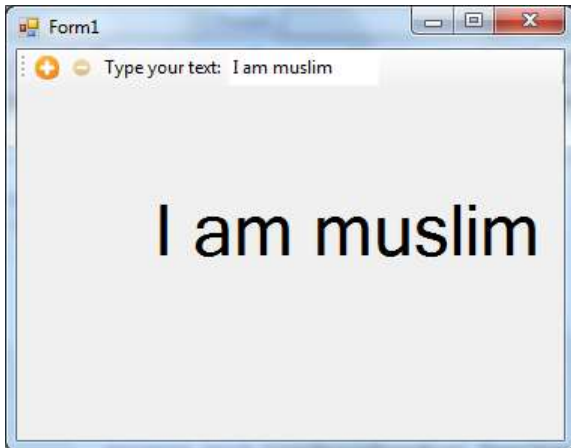
السطر الأول يقوم بتحديد نوع وحجم الخط، وفي السطر الثاني نقوم بزيادة الحجم بنسبة 4، بمعنى أنه كلما ضغط المستخدم على هذا الزر سيزداد حجم الخط في كل مرة بنسبة 4.

بنفس الطريقة اذهب إلى زر النقصان واكتب فيه نفس الشفرة مع استبدال علامة + ب -، لكي يتم نقص حجم الخط:

```
private void toolStripButton2_Click(object sender, EventArgs e)
{
    this.label1.Font = new Font("Tahoma", size);
    size -= 4;
}
```

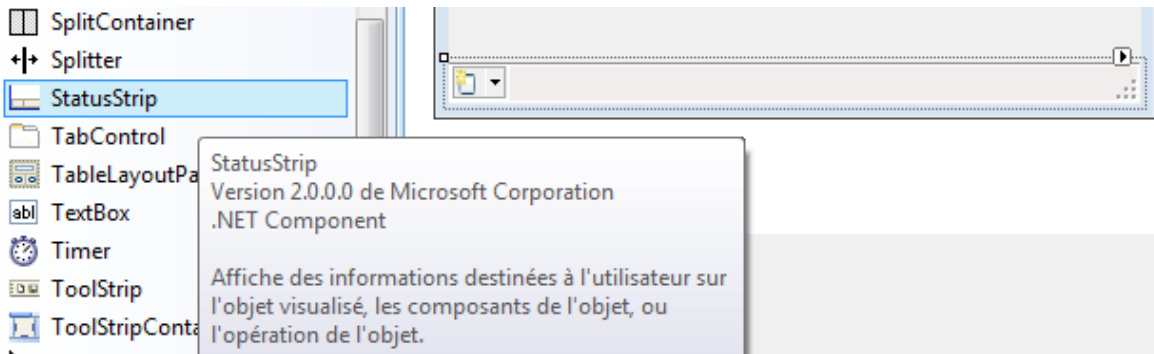
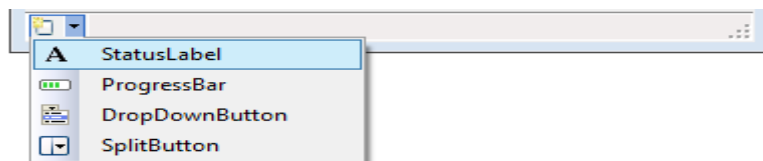
إذا استمررت في الضغط على زر نقص الحجم سيحدث خطأ في البرنامج مفاده أنه لا يوجد حجم خط بقياس 0، لتفادي هذا الخطأ سنتحقق من قيمة المتغير size، فإن كانت تساوي 0 أعدناه إلى نقطة البدء:

```
private void toolStripButton2_Click(object sender, EventArgs e)
{
    if (size == 0)
    {
        size = 8;
    }
    this.label1.Font = new Font("", size);
    size -= 4;
}
```

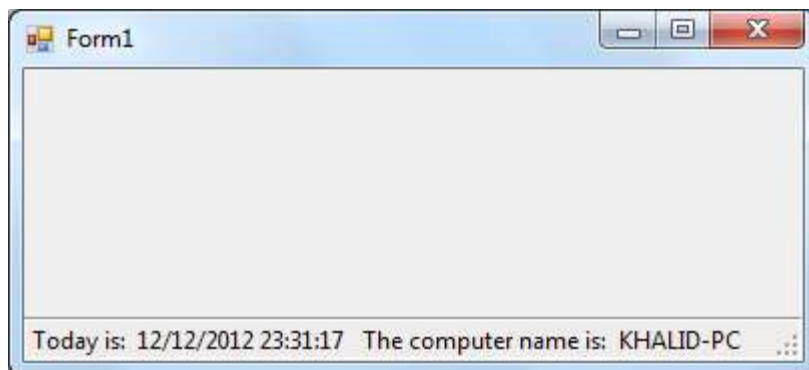
بعد أن تنتهي قم بتنفيذ البرنامج وشاهد النتيجة:

حاول تطوير المثال بطريقتك لتستوعبه أكثر ولا تنس أن تجرب بعض خصائص وأحداث أداة ToolStrip.

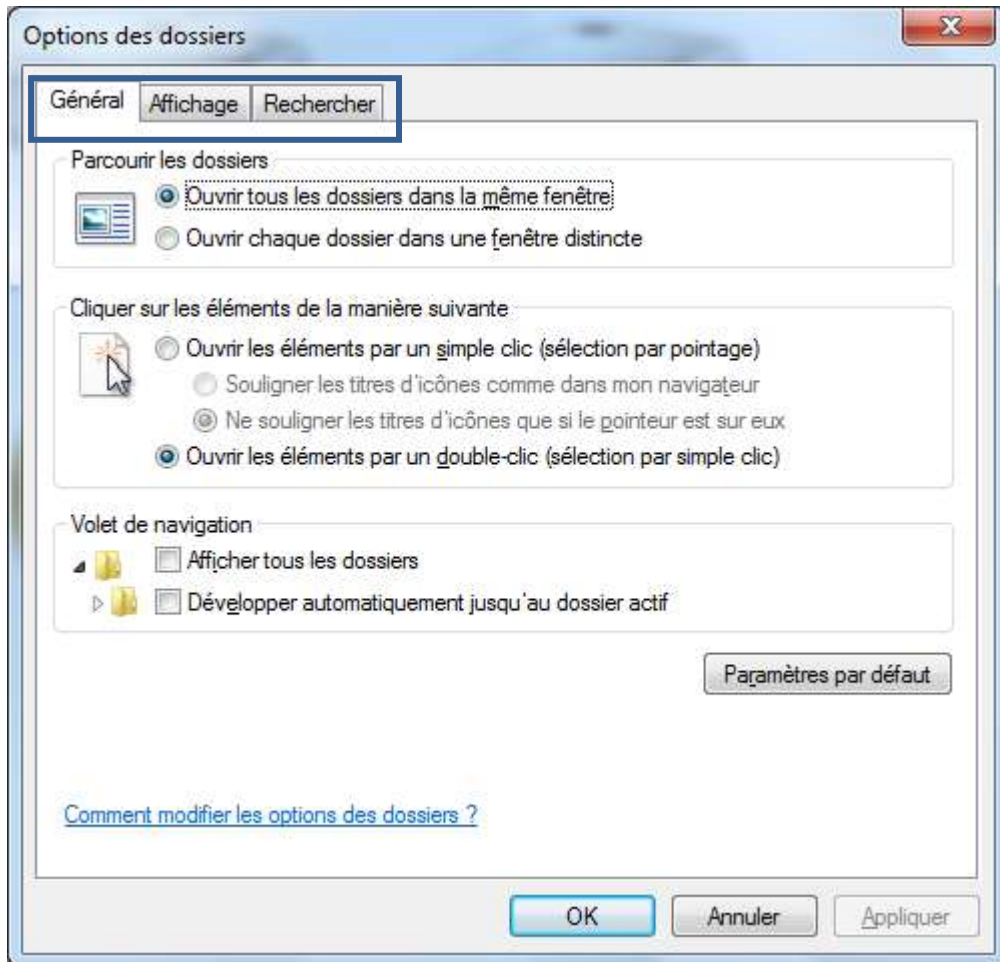
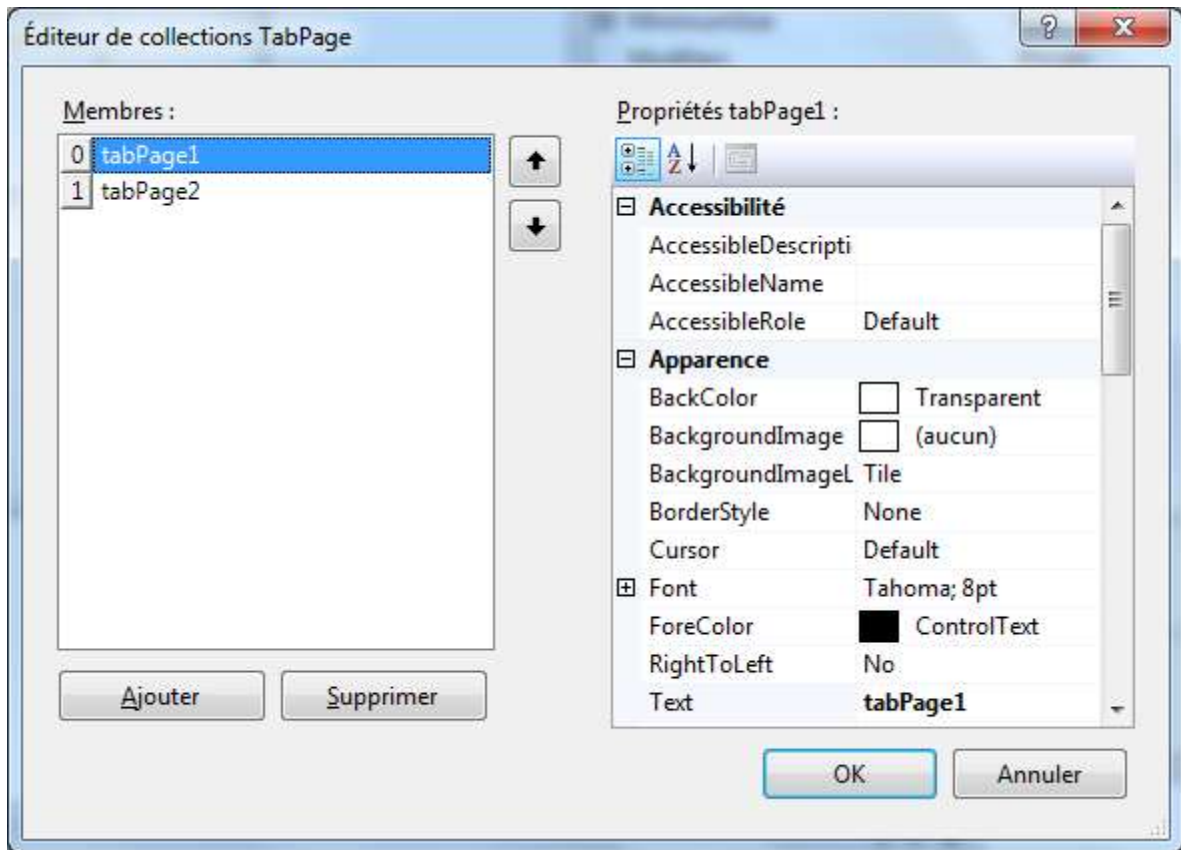
16. أداة شريط الحالة StatusStrip

وتمثل هذه الأداة ذلك الشريط الذي تجده دائما في أسفل نوافذ بعض البرامج، وغالبا ما تكون بغرض عرض بعض المعلومات كالتاريخ الحالي، أو اسم الحاسوب والمستخدم، وأحيانا في برامج الكتابة تكون بغرض إظهار عدد الأسطر وعدد الكلمات والصفحات، كما يبدو في الصورة التالية المأخوذة من برنامج ميكروسوفت وورد Microsoft Word:

سنقوم إن شاء الله بإنجاز تطبيق يعرض تاريخ اليوم، واسم الحاسوب على هذه الأداة، لهذا قم بجذبها من علبة الأدوات إلى الفورم:

قم بالضغط على سهمها الصغير، واختر منه الأداة StatusLabel:

من نافذة الخصائص أعطها الاسم الذي تريد أو اتركها كما هي، ثم انتقل إلى نافذة الكود واذهب تحت الدالة InitializeComponent() واكتب ما يلي :

```
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            this.toolStripStatusLabel1.Text = "Today is:  " +
DateTime.Now;
            this.toolStripStatusLabel1.Text += "   The computer name is:
" + Environment.MachineName;
        }
    }
```

في الأول أظهرنا تاريخ اليوم بالاعتماد على الخاصية Now التابعة للفئة DateTime، وفي السطر الثاني أظهرنا اسم مستخدم الحاسوب عن طريق الدالة MachineName التابعة للفئة Environnement، ستجد بهذه الفئة العديد من الخاصيات التي تمكنك من معرفة بعض التفاصيل كنوع نظام التشغيل وما إلى ذلك، النتيجة ستكون كما يلي:

17. أداة التبويبات TabControl

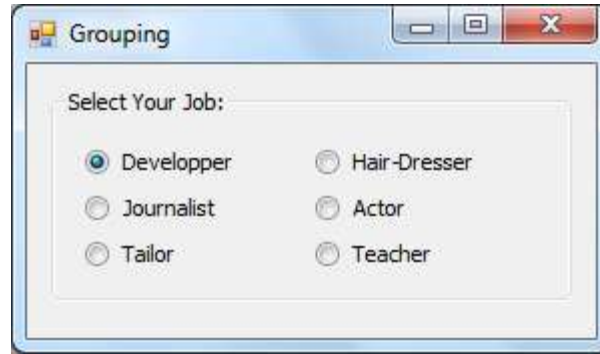
وتستعمل هذه الأداة لتقسيم النافذة إلى مجموعة من الأجزاء، بحيث تستطيع القيام بالعديد من المشاهد Views في نافذة واحدة، كما يبدو في هذه النافذة:

يمكنك إضافة العديد من التبويبات Tabs في نفس النافذة، قم بجذب الأداة TabControl إلى الفورم، ستجد معها تبويبين افتراضيين، يمكنك الزيادة والتعديل في التبويبات عن طريق الخصيصة TabPages، إذا ضغطت عليها ستظهر لك النافذة التالية:

من خلال هذه النافذة تستطيع إضافة تبويبات جديدة وتعديل وحذف التبويبات الموجودة.

لتحديد التبويب الذي تريده أن يظهر أولا، استعمل الدالة SelectTab في نافذة الكود كما يلي:

```
tabControl1.SelectTab(1);
```

بحيث الرقم بين القوسين يمثل ترتيب التبويب المراد إظهاره، علما أن الترتيب يبدأ ب 0، أي أن نتيجة السطر أعلاه ستظهر محتوى التبويب الثاني.

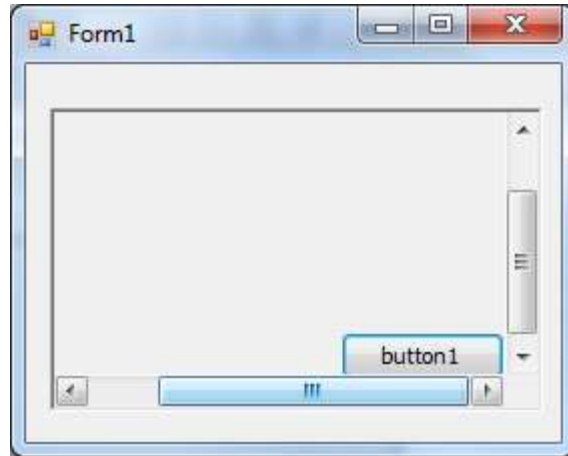
18. أداة التجميع GroupBox

وتستعمل هذه الأداة لتجميع الأدوات داخل مربع، وغالبا ما تستعمل مع أزرار التحديد Radio Button، وتضيف هذه الأداة لمسة جمالية إلى الفورم حيث يبدو أكثر احترافية:

19. لوحة التجميع Panel

مثلها مثل الأداة GroupBox تستعمل لتجميع الأدوات، لكنها لا تظهر عند التنفيذ إلا إذا غيرت قيمة الخصيصة BorderStyle إلى FixedSingle أو Fixed3D، وتفيد هذه الأداة في تطبيق نفس الخصائص على الأدوات المنتمية إليها، كما أن أي عملية تطرأ عليها عند التنفيذ تمس

الأدوات الداخلة فيها، فمثلا لو أردنا إخفاء مجموعة من الأدوات فيكفي تأطيرهم بهذه الأداة ثم تطبيق الخصيصة Visible على الأداة وستختفي كل محتوياتها.

وتتميز هذه الأداة عن GroupBox بكونها تتوفر على أشرطة التمرير العمودية Vertical ScrollBars والأفقية Horizontal Scrollbars، وتستطيع إظهار هذه الأشرطة التمريرية إذا غيرت الخصيصة AutoScroll نحو True، وغيرت الخاصية BorderStyle إلى FixedSingle.

لاحظ بأن هذه الأشرطة لا تظهر إلا إذا كانت الأدوات توجد داخل أداة البانل في مكان لا تستطيع عين المستخدم الوصول إليه، كما تعرض الصورة التالية:

20. علبة الصورة PictureBox

تستعمل هذه الأداة لإظهار الصور، قم بإضافة هذه الأداة إلى الفورم، ثم اذهب إلى خصائصها واختر الخصيصة Image أو BackGroundImage، وقم بجلب صورة من حاسوبك لتظهرها في

الأداة، لتغيير طريقة عرض الصورة اذهب إلى الخصيصة BackGroundImageLayout ، وحدد منها أحد هذه الاختيارات:

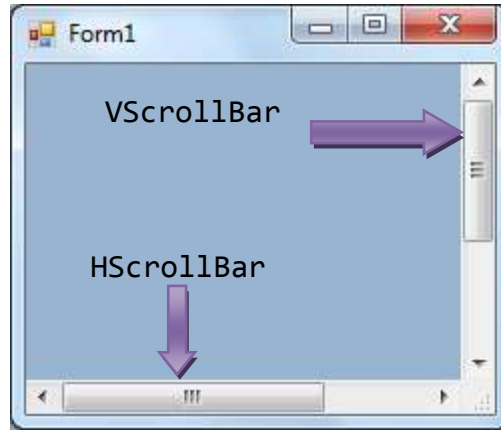
- None: ستظهر الخلفية في أعلى يسار الفورم.
- Tile: ستظهر الصورة مكررة طولا وعرضا.
- Center: ستظهر الصورة في وسط الأداة.
- Stretch: ستظهر الصورة ممدة على كل الفورم.
- Zoom: ستظهر الصورة في أقصى مقاسها مركزة في وسط الفورم.

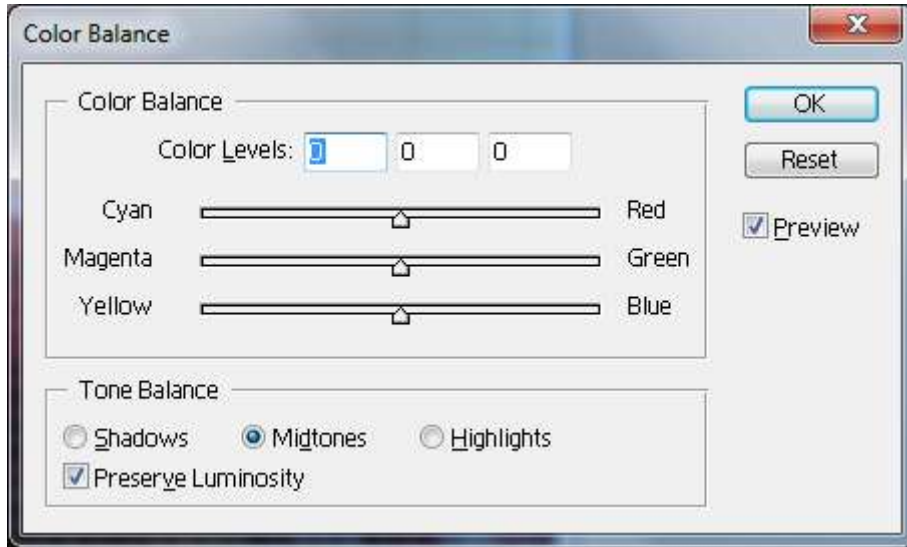
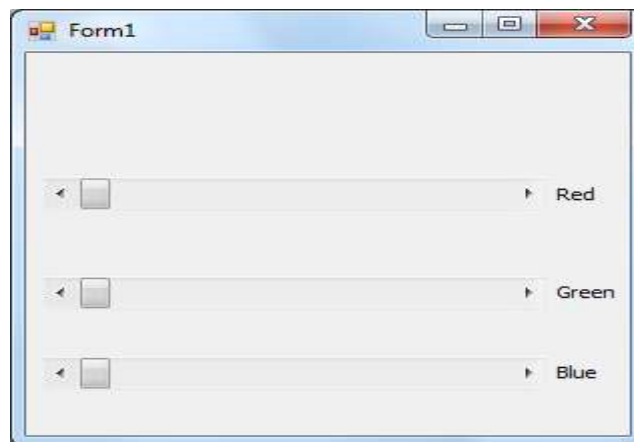
ويمكنك إضافة صورة إلى الأداة PictureBox عن طريق الكود أيضا بواسطة الدالة FromFile التابعة للفئة Image:

```
this.pictureBox1.Image = Image.FromFile("MyPic.JPG");
```

21. شريط التمرير ScrollBar

هنالك أشرطة تمرير عمودية VScrollBar، وهي تلك التي تراها دائما على يمين صفحات الويب، ويكون دورها هو السماح بالنزول من أعلى إلى تحت والعكس أيضا، وهنالك أشرطة تمرير أفقية HScrollBar وتظهر إذا زدت في حجم صفحة الويب، بحيث تشاهد أشرطة على أسفل الصفحة

تمكنك من الاتجاه من اليسار إلى اليمين والعكس أيضا، ولعلك رأيت صورة هذه الأشرطة حينما تحدثنا عن أداة التجميع Panel، وهذه صورة للأداتين VScrollBar و HScrollBar:

الغاية من الشريطين ليست دائما التمرير، بل يتجاوزان هذا الأمر إلى شيء آخر مهم، وهو التلاعب بقيم أداة ما زيادة ونقصانا مثل دور الأداة الموالية TrackBar، سنعرض الآن نموذجا بأداة HScrollBar نقوم من خلاله بتغيير لون خلفية الفورم بالتدريج.

هذه النافذة مأخوذة من برنامج Adobe Photoshop تقوم بتغيير لون الصورة بواسطة الخلط بين الألوان الثلاثة (الأحمر، الأزرق، الأخضر)، الحديث عن الألوان حديث طويل لأننا سنخوض في الأنظمة الست عشرية، وماهو المجال الرقمي للألوان وما إلى ذلك، مما لا يتسع له المقام، على العموم شاهد هذه الصورة وإن كنت من مستخدمي برنامج الفوتوشوب ستعرفها أكثر:

سنحاول إنجاز نافذة مثل هذه يقوم المستخدم فيها بتغيير لون خلفية الفورم عن طريق تغيير قيم الأشرطة الأفقية، في الصورة أعلاه الأداة المستخدمة هي TrackBar، ولكننا سنستعمل أداة HScrollBar لأننا بصدد الحديث عنها، طيب قم بإنجاز فورم مماثل لما في الصورة، أو بكل بساطة كهذا الفورم:

بعد أن تضيف الأشرطة الأفقية الثلاثة، قم بتغيير أسمائهم ليسهل استخدامهم، وكذلك قم بتغيير الخصيصة Minimum لكل أداة وضع فيها القيمة 0، وغير الخصيصة Maximum إلى 250، لأن هذا هو مجال الألوان.

الحدث الذي ينفذ عند تغيير قيمة الشريط هو Scroll، ويمكنك الولوج إليه عن طريق تحديد كل أداة والذهاب إلى نافذة الأحداث (طبعا هي نافذة الخصائص بعد الضغط على أيقونة الأحداث)، وستجد هذا الحدث هناك، اضغط عليه مرتين لتدخل إليه، واكتب فيه هذا الكود:

```
        private void RedSB_Scroll(object sender, ScrollEventArgs e)
        {
            this.BackColor = Color.FromArgb(RedSB.Value, GreenSB.Value,
BlueSB.Value);
        }
```

الدالة FromArgb تقوم بتكوين اللون من القيم الرقمية للبرامترات الثلاثة، وفي كل مرة تتغير القيمة الرقمية يتغير اللون، فمثلا إذا كتبنا هكذا :

```
this.BackColor = Color.FromArgb(1, 1, 1);
```

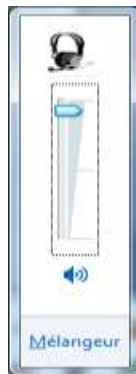
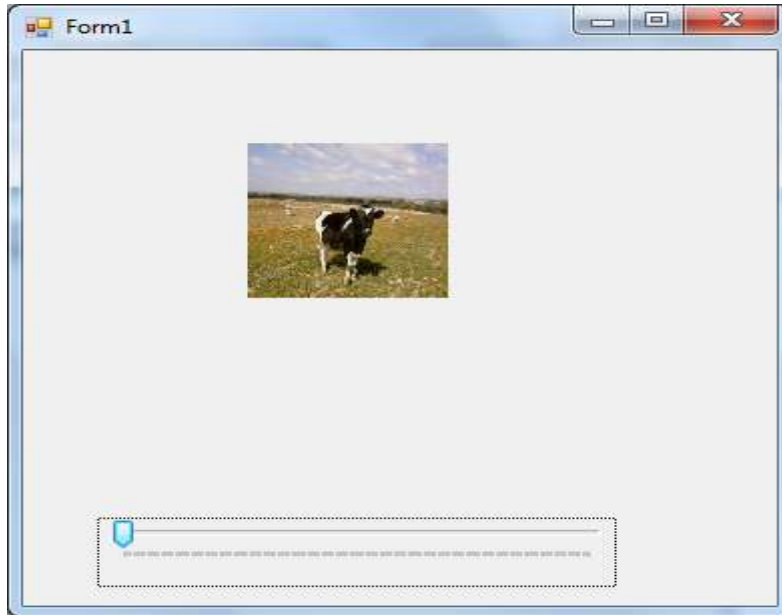
فسيظهر لون خلفية الفورم وهو أسود، وكلما غيرنا القيمة تغير اللون، وهذا ما كتبناه في هذا السطر:

```
this.BackColor = Color.FromArgb(RedSB.Value, GreenSB.Value, BlueSB.Value);
```

قم بكتابة هذا السطر في الحدث Scroll الخاص بالشريطين المتبقيين، ثم نفذ لتشاهد النتيجة:

قم بتمرير الأشرطة وشاهد ماذا سيحدث

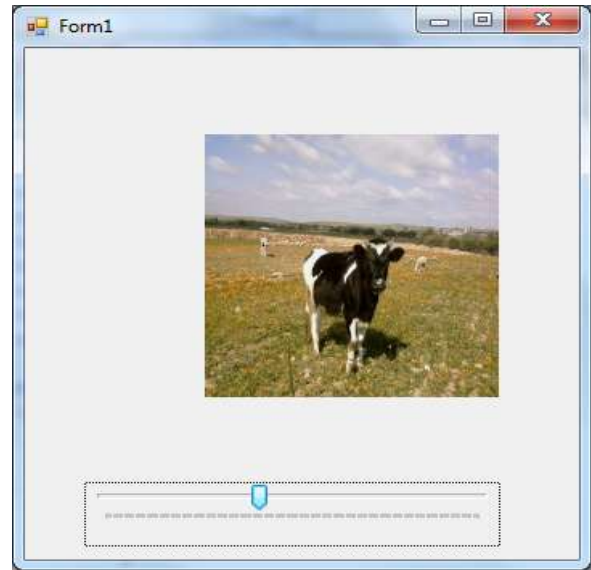
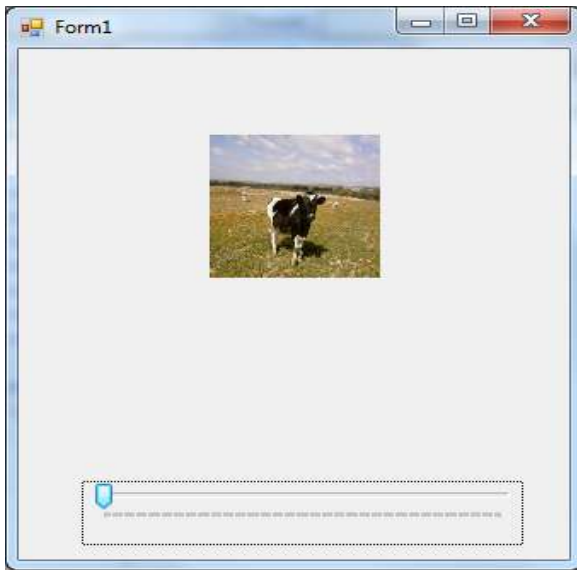
22. شريط التدرج TrackBar

تحدثنا عنه قبل قليل وشاهدناه في نافذة تابعة لبرنامج الفوتوشوب، وغالبا ما تستعمل هذه الأداة للتحكم في قيمة شيء ما بالزيادة والنقصان، كما يظهر في الصورة أسفله التي تعرض متحكم الصوت في الويندوز وهو يستعمل هذه الأداة عموديا:

سننجز الآن تطبيقا بسيطا يقوم بزيادة ونقصان حجم صورة معينة بواسطة هذه الأداة، وليكن الفورم بعد سحب الأداتين TrackBar و PictureBox عليه كما يلي:

قم بإضافة صورة إلى pictureBox1، ولتكن بالأخص صورة بقرة وإلا فلن يشتغل البرنامج 😜

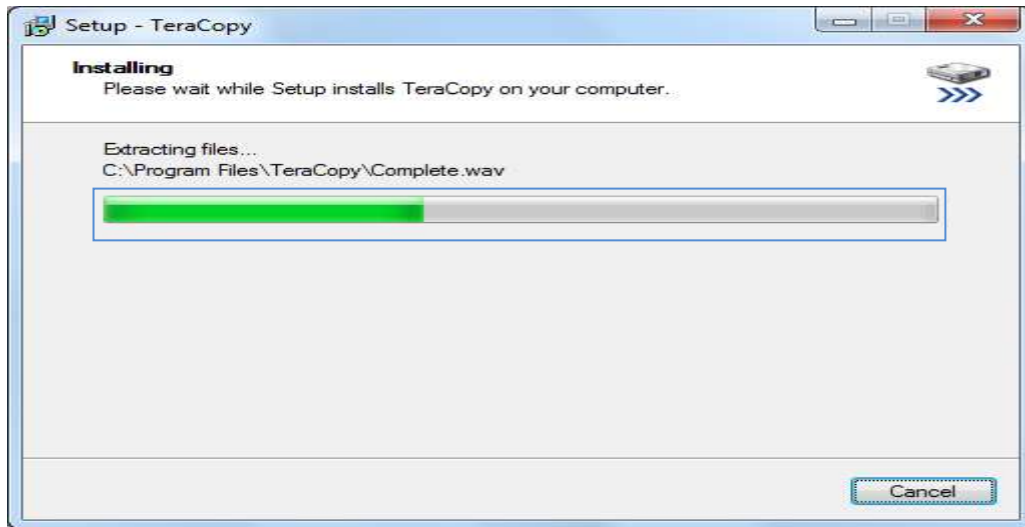
احذر أن تصدقني فأنا أمزح، طيب لنعد إلى الجد 😃، قم بتحديد الأداة trackBar1 واذهب إلى نافذة الخصائص وقم بتغيير قيم الخصائص Minimum وMaximum كما تريد لكن يفضل أن تكون قيمة الخاصية Minimum تساوي حجم Size أداة pictureBox1، بعد أن تعدل في الخصائص كما تحب، اذهب إلى الحدث Scroll الخاص بالأداة trackBar1 عن طريق الضغط عليها مرتين، ثم اكتب السطر التالي:

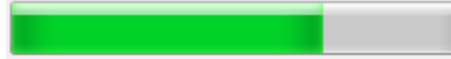
```
private void trackBar1_Scroll(object sender, EventArgs e)
{
       this.pictureBox1.Size = new Size(trackBar1.Value, trackBar1.Value);
}
```

قمنا بتحديد طول وعرض أداة الصورة وفقا لقيمة شريط التدرج trackBar1.

نفذ وغير قيمة شريط التدرج لتشاهد النتيجة:

طبعا الغرض من التطبيق هو تعريف الأداة، حاول أن تعدل المثال بطريقتك، كأن تصنع عارض صور.

23. شريط التطور ProgressBar

وتظهر هذه الأداة حينما نرغب في جعل المستخدم ينتظر قليلا، وغالبا ما تراها في برامج التحميل والتنصيب وعند نسخ الملفات وحذفها، كما تعرض الصورة التالية:

وتتوفر هذه الأداة على الخصائص Minimum و Maximum التي تحدد أدنى وأقصى قيمة لتقدم الأداة، ويمكننا زيادة القيمة عن طريق الكود بواسطة الدالة Increment أو بواسطة value وسوف نتعرف على كيفية جعلها تتغير تلقائيا في الفقرة القادمة مع الأداة Timer.

تتوفر هذه الأداة على الخصيصة Style التي تمكنك من تغيير طريقة التقدم:

:Continuous

24. أداة العداد Timer

أحيانا تحتاج إلى تنفيذ بعض المهام عند التنفيذ Run Time، كإظهار الوقت الحالي، أو تغيير قيمة معينة مع مرور الوقت، لفعل ذلك تعال بنا نتعرف على هذه الأداة الجميلة.

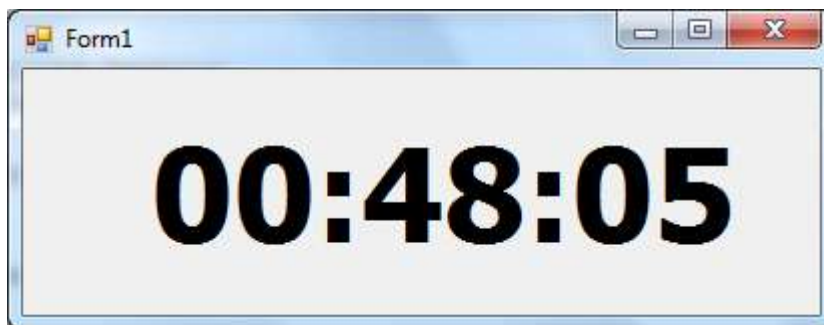
قم بجذب هذه الأداة إلى الفورم، و ألق نظرة على خصائصها القليلة، ستجد ضمنها خاصيتين مهمتين وهما:

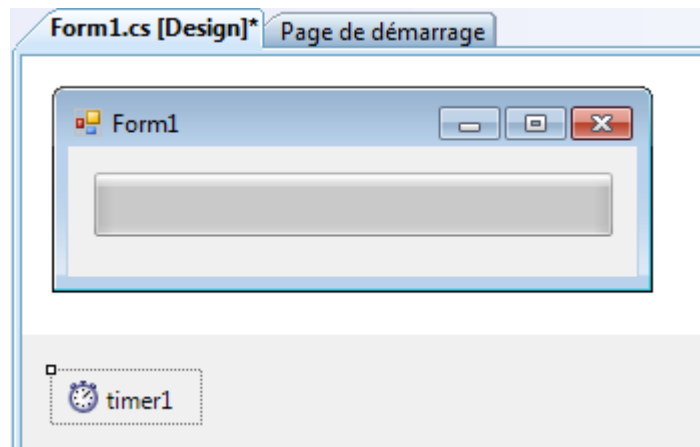
الخصيصة	دورها
Enabled	وتأخذ قيمة منطقية إما True أو False، لجعل الأداة تشتغل غير القيمة نحو True
Interval	نكتب فيها مقدار الوقت الذي نريد أن يتكرر فيه الحدث بالوحدة : جزء الثانية MilliSecond مع العلم أن: 1 second=1000 MilliSecond

بعد أن تجعل الخصيصة Enabled تساوي True، وتضع في الخصيصة Interval القيمة 1000، أضف أداة Label إلى الفورم، ثم حدد العداد timer1 واضغط عليه مرتين لتنتقل إلى الحدث Tick المتعلق به، ثم اكتب السطر التالي:

```
private void timer1_Tick(object sender, EventArgs e)
{
    this.label1.Text = DateTime.Now.ToLongTimeString();
}
```

الدالة ToLongTimeString، تقوم بعرض الوقت كاملا، ستجد العديد من دوال الوقت كل واحدة تظهر شكلا مختلفا.

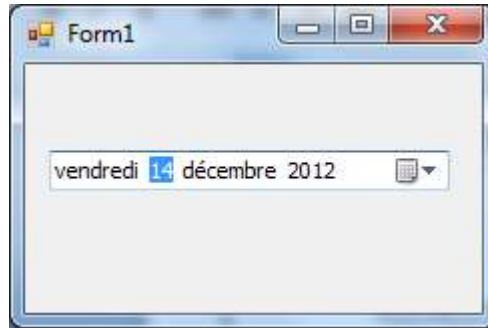
إذا قمت بتنفيذ البرنامج ستطالعك النتيجة التالية، وستشاهد بأن قيمة الوقت تتغير كل ثانية، لأننا أعطينا الخصيصة Interval القيمة 1000 التي قلنا بأنها تعادل ثانية واحدة:

أتمنى أن تنجح في إنجاز هذا التطبيق، الآن سننشئ تطبيقا جديدا يقوم بجعل أداة ProgressBar تتغير حسب المجال Interval الخاص بالعداد Timer، وأنا متأكد بأنك ستحتاج هذا الأمر في وقت قادم حينما تبدأ في تطوير برامج احترافية.

أضف إلى الفورم الجديد أداة ProgressBar وأداة Timer، ثم قم بتفعيل الخصيصة Enabled وإعطاء المجال Interval الذي تريد:

قم بالدخول إلى الحدث Tick الخاص ب timer1 واكتب الكود التالي:

```
private void timer1_Tick(object sender, EventArgs e)
{
    while (progressBar1.Value < progressBar1.Maximum)
    {
        progressBar1.Value += 2;
    }
}
```

نقوم بزيادة تقدم قيمة الأداة progressBar1 مادامت القيمة لم تصل إلى القيمة القصوى، جرب وشاهد بنفسك، تستطيع تغيير القيمة القصوى لل progressBar1 عن طريق نافذة الخصائص، كما تستطيع تأخير عملية التحميل Loading بالزيادة في مجال Interval العداد timer1، احذر أن تنس تفعيل الخاصية Enabled وإلا فلن يشتغل العداد وبالتالي لن يحدث أي شيء.

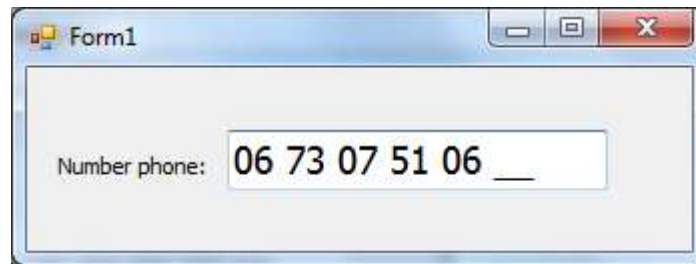
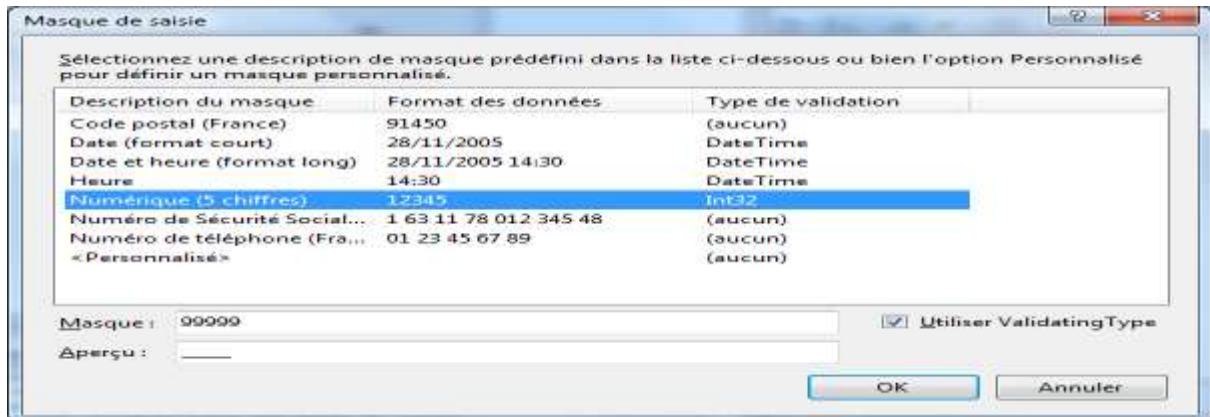
25. أداة التاريخ DateTimePicker

تتيح هذه الأداة للمستخدم اختيار تاريخ معين، وتكون قيمتها الافتراضية هي تاريخ اليوم، وهذه صورة للأداة بعد جذبها إلى الفورم:

حينما يضغط المستخدم على سهم الأداة، سيظهر له تفصيل لأيام الشهر الحالي، ويمكنه التنقل بين الأيام والشهور والأعوام:

ويمكنك معرفة التاريخ الذي اختاره المستخدم بواسطة الخاصية value التابعة للأداة:

```
DateTime mydate = this.dateTimePicker1.Value;
```

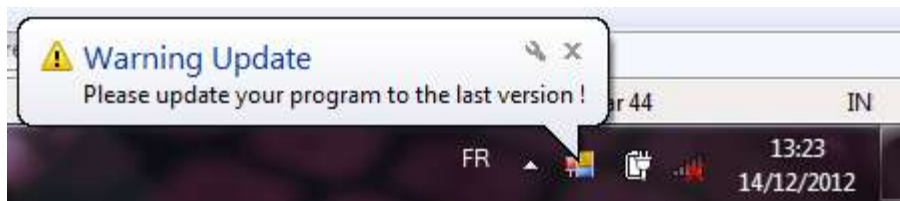
26. أداة علبة النص المجهزة MaskedTextBox

وهي شبيهة بعلبة النص العادية TextBox، إلا أنها تكون على شكل قالب معين، فمثلا حينما تريد فرض كيفية كتابة رقم الهاتف على المستخدم أو عنوانه البريدي، فسوف تحتاج إلى هذه الأداة، انظر إليها أولا:

بحيث تصبح الأداة قابلة للكتابة وفق القالب المحدد لها فقط، ويمكنك تغيير القالب عن طريق الذهاب إلى نافذة خصائص الأداة واختيار الخصيصة Mask، لكي تظهر لك النافذة التالية:

ويمكنك كتابة القالب الذي تريد، ولكي أكمل الشرح على أحسن وجه سأقدم لك جدول الرموز الممكن كتابتها عند تصميم القالب ودور كل رمز:

الرمز	دوره
0	الحروف مرفوضة، خاص بكتابة الأرقام فقط.
9	الحروف مرفوضة ماعدا الفراغ Blank، يسمح بكتابة الأرقام.
#	الحروف مرفوضة ماعدا الفراغ والرمزين + و -، يسمح بكتابة الأرقام.
L	الأرقام مرفوضة، خاص بكتابة الحروف.
?	الأرقام مرفوضة، يسمح بكتابة الحروف.
&	الأرقام والحروف مقبولة.
C	الأرقام والحروف مقبولة إضافة إلى الرموز % & *
A	الأرقام والحروف مقبولة، والرموز % * & مرفوضة.
.	الفاصلة العشرية.
,	الفاصلة بين الآلاف.
:	الفاصل بين الأوقات، تجده بين الساعات

	والدقائق والثواني.
/	الفاصل بين التواريخ.
$	رمز نقدي.
<	تحويل الحروف التي تلي هذا الرمز إلى الحالة الصغيرة Lower Case
>	تحويل الحروف التي تلي هذا الرمز إلى الحالة الكبيرة Upper case
|	إيقاف تحويل الحروف الكبيرة والصغيرة.

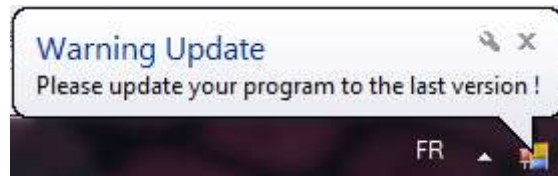
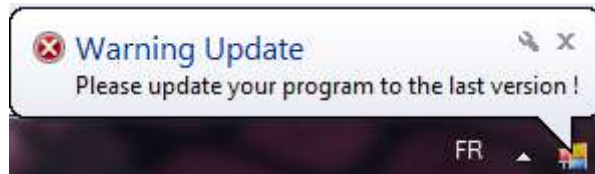
27. مؤشر الأيقونات NotifyIcon

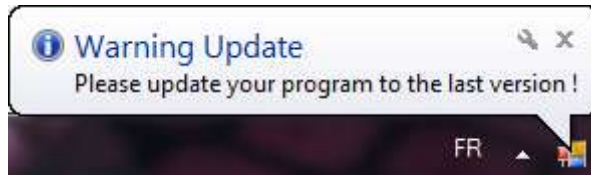
تستعمل هذه الأداة لإظهار أيقونة البرنامج على يمين شريط المهام قرب التاريخ والوقت، ويكون الغاية من ذلك هي جعل البرنامج في حالة اشتغال حتى لو قمت بإغلاق نافذته، وكذلك لإظهار رسائل ومعلومات للمستخدم، كما تظهر النافذة التالية:

لنتعرف أكثر على هذه الأداة، قم بإضافتها إلى الفورم، ثم اذهب إلى نافذة الكود واكتب فيها هذه الأسطر مباشرة بعد الدالة InitializeComponent:

```
            notifyIcon1.Icon = this.Icon;
            notifyIcon1.BalloonTipIcon = ToolTipIcon.Warning;
            notifyIcon1.BalloonTipText = "Please update your program to
the last version !";
            notifyIcon1.BalloonTipTitle = "Warning Update";
            notifyIcon1.ShowBalloonTip(12);
```

الخصيصة Icon التابعة للأداة notifyIcon1، تقوم بتحديد الأيقونة المراد إظهارها في الرسالة، في حالتنا هذه جعلناها نفس أيقونة الفورم.

وفي السطر الثاني حددنا نوع الرسالة BalloonTipIcon وتتوفر هذه الخصيصة على ثلاث اختيارات:

- None: لا تظهر في الرسالة أية أيقونة.

- Error: تظهر الرسالة على شكل رسالة تنبيهية إلى وجود خطأ

- Warning: تظهر رسالة تحذيرية كما عرضنا في الصورة في المثال الأول.

Info: تظهر رسالة إخبارية مفادها إعلام المستخدم بشيء ما.

إذن فالخصيصة BalloonTipIcon تقوم بتحديد مظهر الرسالة أتحذيرية هي أم إخبارية.

أما السطر التالي:

```
notifyIcon1.BalloonTipText = "Please update your program to the last version !";
```

فيقوم بتحديد نص الرسالة.

وفيما يخص السطر التالي:

```
notifyIcon1.BalloonTipTitle = "Warning Update";
```

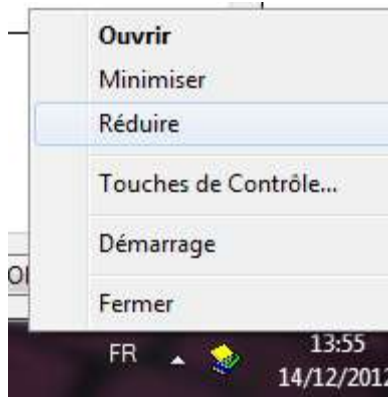
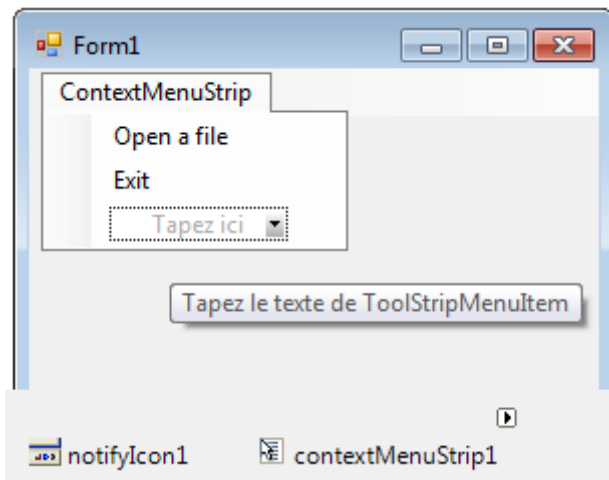
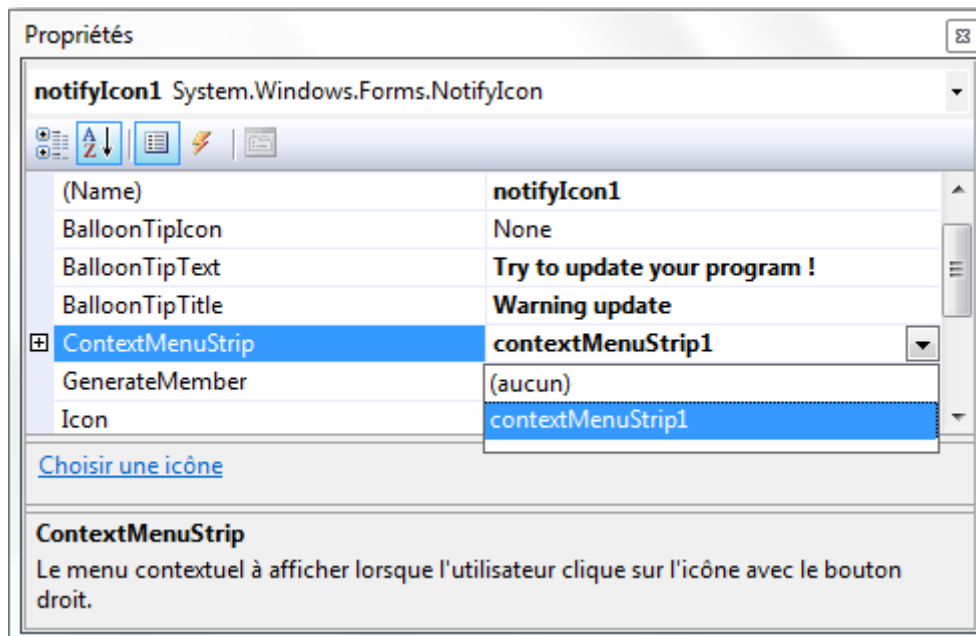
فدوره هو تحديد عنوان الرسالة.

وفي الأخير، السطر التالي:

```
notifyIcon1.ShowBalloonTip(12);
```

هذه الدالة هي التي تقوم بإظهار الرسالة، أما القيمة الرقمية 12 فهي من أجل إعطاء مدة ظهور الرسالة، وهي بوحدة الجزء من الثانية µs، ويمكنك جمع الأسطر السابقة الخاصة بالأيقونة والنص والعنوان في هذه الدالة على الشكل التالي:

```
notifyIcon1.Icon = this.Icon;
notifyIcon1.ShowBalloonTip(12, "Warning Update", "Update Your..", ToolTipIcon.Error);
```

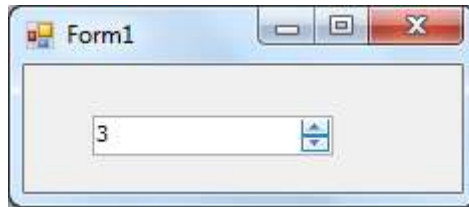
الآن سوف نرى كيفية الربط بين هذه الأداة NotifyIcon وبين الأداة ContextMenuStrip التي رأيناها سابقا، كما يعرض أحد البرامج:

عند الضغط بيمين الماوس على أيقونة البرنامج ستظهر لك قائمة ContextMenuStrip، سننجز تطبيقا مشابها، قم بإضافة الأداتين ContextMenuStrip و NotifyIcon إلى الفورم، ثم ضع في عناصر القائمة ContextMenuStrip ما تشاء :

بعد ذلك اربط بين الأداتين كما رأينا سابقا مع بعض الأدوات (اذهب إلى خصائص الأداة notifyIcon1 واختر الخصيصة ContextMenuStrip ثم حدد ContextMenuStrip1 ، كما تعرض الصورة التالية)

جيد، الآن قمنا بالربط بين الأداتين، ادخل إلى نافذة الكود واكتب بنفسك الكود السابق الخاص بإظهار أيقونة الفورم على شريط المهام وهو كما يلي :

```
        public Form1()
        {
            InitializeComponent();
            notifyIcon1.Icon = this.Icon;
            notifyIcon1.Visible = true;
        }
```

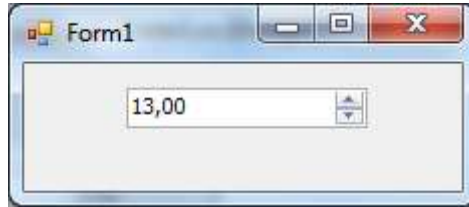
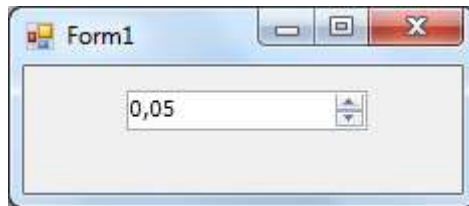
احذر أن تنس كتابة السطر الثاني .

قم بالتنفيذ، واذهب إلى شريط المهام واضغط على أيقونة البرنامج بيمين الماوس لتطالعك النتيجة التالية:

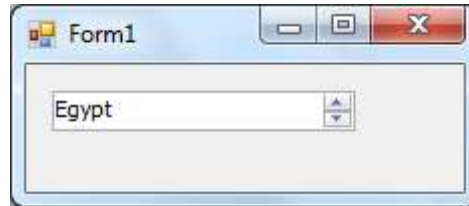
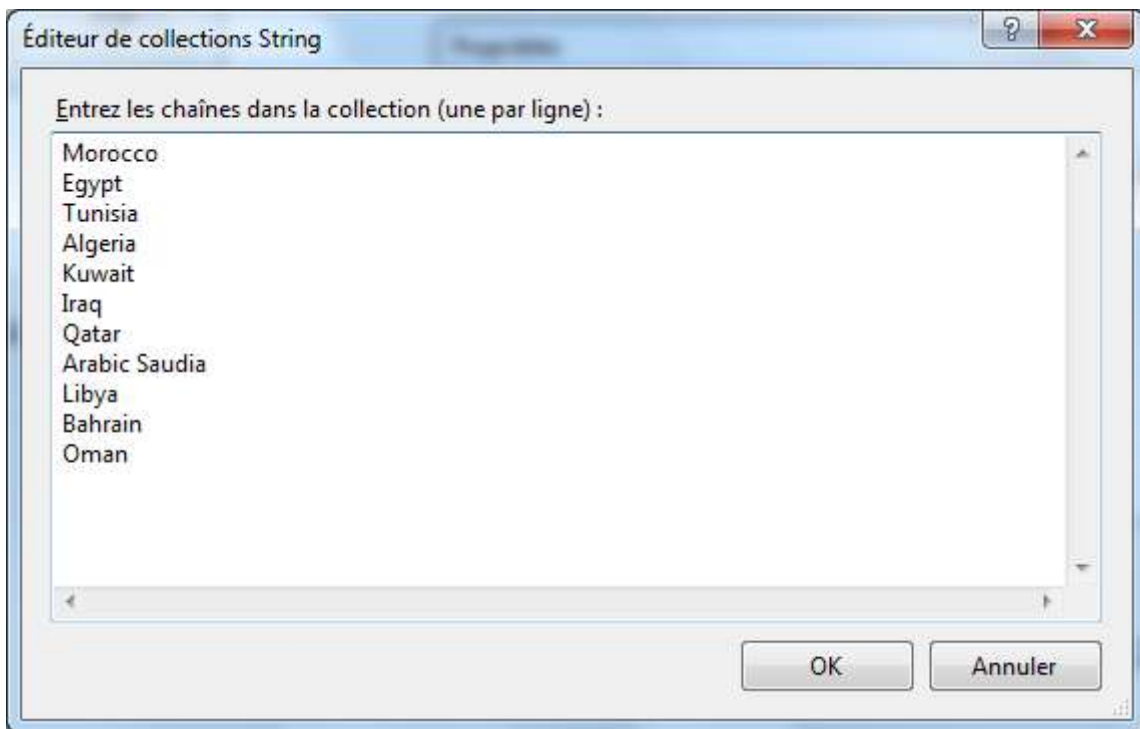
28. أداة الأرقام التدرجية NumericUpDown

تقوم هذه الأداة بعرض قيم رقمية مع إمكانية تغيير القيمة تصاعديا وتنازليا، وهذه صورة للأداة لتتعرف عليها:

وتتوفر على الخصائص Minimum و Maximum لتحديد القيمة الدنيا والقصوى للأداة، ويمكننا معرفة القيمة الرقمية التي حددها المستخدم عن طريق الخاصية Value وهي من نوع عشري decimal:

```
decimal number = numericUpDown1.Value;
```

و تضم هذه الأداة أيضا الخصيصة DecimalPlaces، التي تسمح بتحديد عدد الأرقام بعد الفاصلة، بمعنى لو أعطيناها القيمة 2، فسوف يظهر رقمان بعد الفاصلة العشرية، كما توضح الصورة التالية:

ستلاحظ بأن قيمة الأعداد ستبقى صحيحة طبيعية Integer، إذا أردت جعلها عشرية فما عليك سوى الذهاب إلى الخصيصة Increment ووضع القيمة 0,01 فيها وسلاحظ أن معدل التغير صار عشريا:

الحدث الذي يتولد عندما يقوم المستخدم بتغيير قيمة الأداة numericUpDown1 هو ValueChanged.

29. أداة النصوص التدرجية DomainUpDown

وهي نظيرة الأداة السابقة ولكن لعرض النصوص، وهذه صورها:

لتعبئة الأداة بالبيانات، قم بتحديدها ثم اختر الخصيصة Items وقم بكتابة كل عنصر في سطر:

تستطيع فعل ذلك بالكود أيضا، كما يلي:

```
this.domainUpDown1.Items.Add("my First item");
```

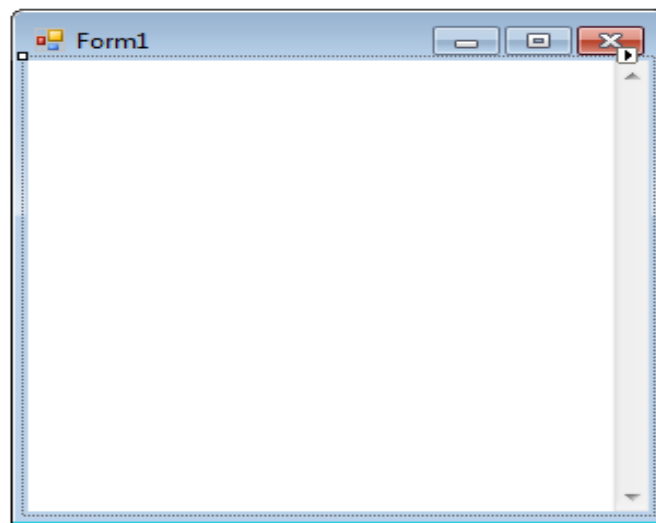
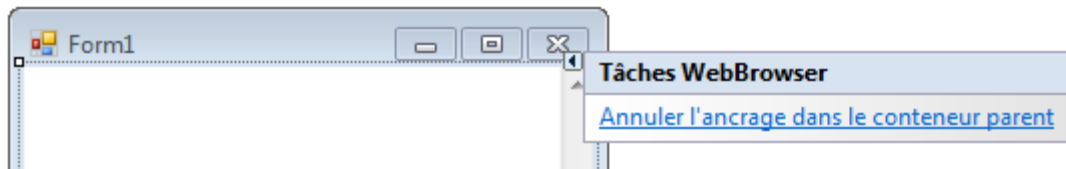
لجعل أحد العناصر يظهر أولا، اذهب إلى الخصيصة SelectedIndex وقم بكتابة رتبة هذا العنصر رقميا:

```
this.domainUpDown1.SelectedIndex = 5;
```

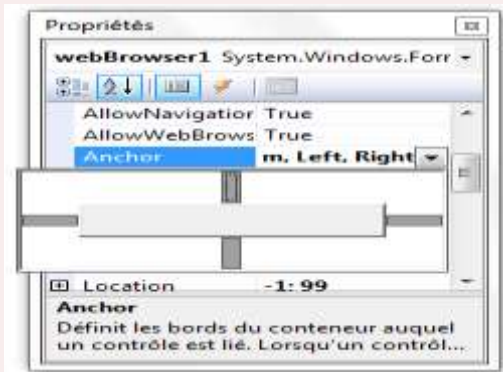
الحدث SelectedItemChanged يتولد في كل مرة يقوم المستخدم بتغيير العنصر المحدد.

30. أداة متصفح الويب WebBrowser

وتستعمل هذه الأداة لصنع متصفح الأنترنت مثل Internet Explorer و Google Chrome، وتستطيع بواسطة هذه الأداة عرض الملفات النصية أيضا، فضلا عن صفحات الويب.

قم بجذب أداة WebBrowser إلى الفورم، لاحظ كيف تقوم بالتمدد على كل الفورم:

تستطيع إلغاء التمدد وتغيير مقاس الأداة، عن طريق ضغط السهم الموجود أعلى الأداة واضغط على الأمر الموجود هناك:

سنقوم بتصميم المتصفح التالي لنستوعب هذه الأداة جيدا:

الأدوات التي يضمها التطبيق:

الأداة	دورها
Label	لعرض اسم المتصفح
Button	للرجوع إلى الصفحة السابقة.
Button	للذهاب إلى الصفحة الموالية.
TextBox	لكتابة رابط الموقع المراد الذهاب إليه. ويمكنك تعويض هذه الأداة بأداة MaskedTextBox بعد أن تحدد لها قالبا لا يسمح سوى بكتابة روابط الويب.
Button	لتأكيد الدخول على الموقع المكتوب.
WebBrowser	لعرض محتوى الموقع ولتصفحه. واجعل الخصيصة Anchor الخاصة به على الزوايا الأربعة، لكي يتغير حجم المتصفح وفق حجم الورم:

بعد أن تنهي التصميم، اذهب إلى الزر Go وادخل إلى الحدث Click الخاص به واكتب فيه هذا السطر الذي يقوم بالذهاب إلى الصفحة المكتوبة في علبة النص:

```
webBrowser1.Navigate(textBox1.Text);
```

الدالة Navigate تقوم بفتح الموقع أو الملف الذي يكتب عنوانه في برامترها.

ادخل إلى الحدث Click الخاص بزر الرجوع إلى الوراء، واكتب فيه ما يلي:

```
webBrowser1.GoBack();
```

نفس الأمر فيما يخص زر التقدم إلى الأمام، واكتب السطر التالي:

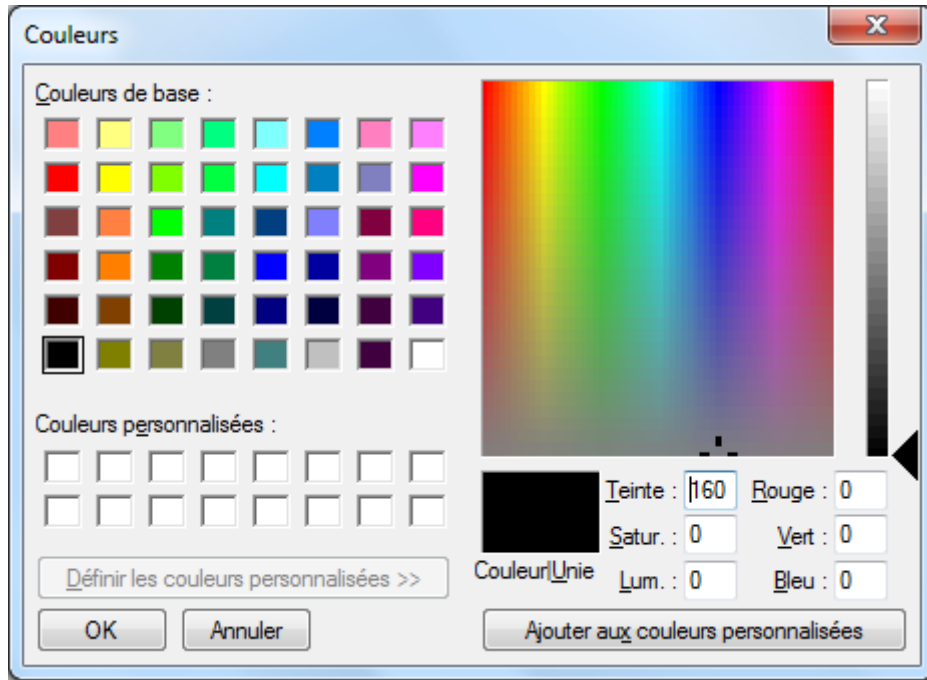
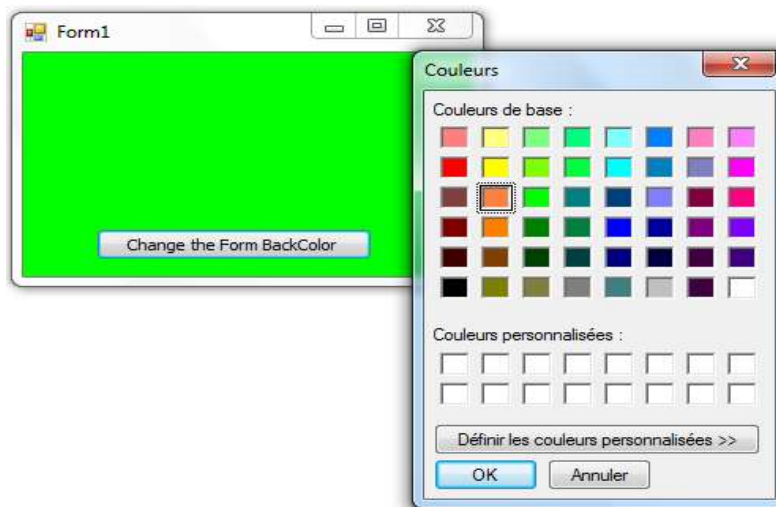
```
webBrowser1.GoForward();
```

قم بالتنفيذ وشاهد النتيجة وهنيئا لك تصميم متصفح خاص بك، ستجد العديد من الدوال والخصائص التي تؤدي مجموعة من المهام كالعودة إلى الصفحة الرئيسية GoHome، أو تحديث الصفحة Refresh.

31. أداة اختيار الألوان ColorDialog

تقوم هذه الأداة بفتح علبة الألوان للمستخدم ليختار منها ما يشاء، ويمكننا اختيار اللون بواسطة الخصيصة Color التابعة لهذه الأداة، كما يبدو في هذا السطر:

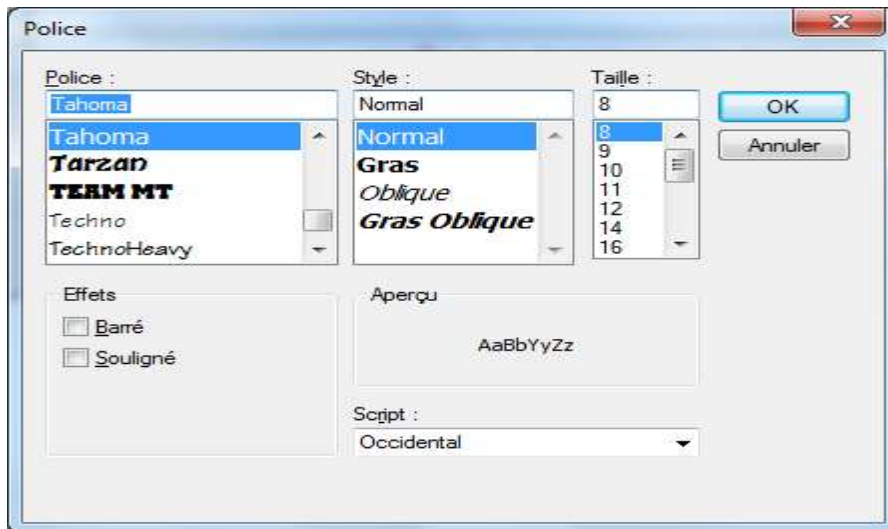
```
this.BackColor = colorDialog1.Color;
```

وهذه صورة الأداة ColorDialog التي نتحدث عنها:

سنقوم بإنجاز تطبيق بسيط، يضم هذه الأداة ColorDialog وزر Button، عند الضغط على هذا الزر سنظهر علبة الألوان ليختار منها المستخدم اللون الذي يريد تطبيقه على خلفية الفورم:

بعد أن تضيف الأداتين (ColorDialog ،Button) إلى الفورم، قم بالدخول إلى الحدث Click التابع للزر، واكتب فيه ما يلي:

```
        private void button1_Click(object sender, EventArgs e)
        {
            colorDialog1.ShowDialog();
            this.BackColor = colorDialog1.Color;
        }
```

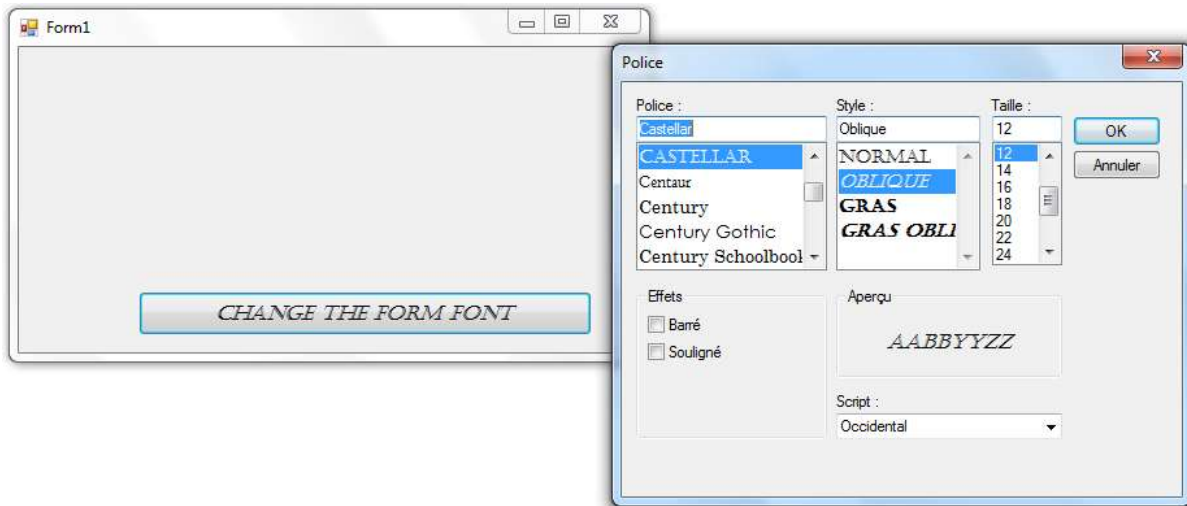
الإجراء ShowDialog يظهر نافذة الألوان، وفي السطر الثاني نقوم بتغيير لون الفورم وفق اللون المحدد من قبل المستخدم، جرب هذا المثال وشاهد النتيجة.

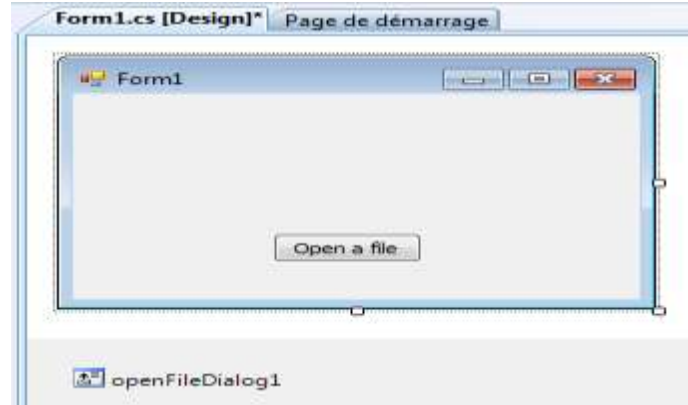
32. أداة اختيار الخطوط FontDialog

طريقة اشتغالها شبيهة بالأداة السابقة، وتستعمل هذه الأداة لتغيير خصائص الخط Font، كما تعرض النافذة التالية:

قم بإنشاء فورم شبيه لما في التمرين السابق، وقم بالولوج إلى الحدث Click التابع للزر واكتب فيه ما يلي:

```
        private void button1_Click(object sender, EventArgs e)
        {
            fontDialog1.ShowDialog();
            this.Font = fontDialog1.Font;
        }
```

الخاصية Font هي المسؤولة عن تغيير الخط، أما السطر الأول فهو يقوم بإظهار علبة الخط.

إذا جربت المثال ستحصل على نتيجة كهذه:

33. أداة فتح الملفات OpenFileDialog

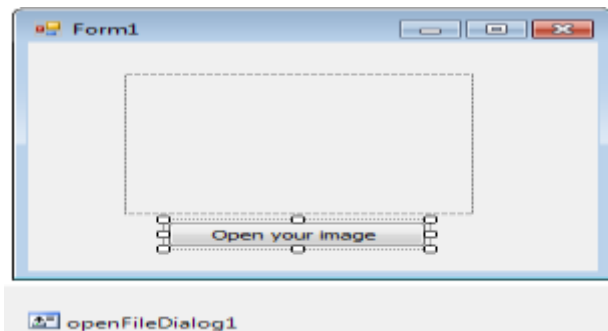
تستعمل هذه الأداة لفتح ملف أو مجموعة من الملفات، قم بإضافة الأداة openFileDialog إلى الفورم، وأضف أيضا إليه زرا لفتح الأداة:

ادخل إلى الحدث Click الخاص بالزر، واكتب السطر التالي:

```
openFileDialog1.ShowDialog();
```

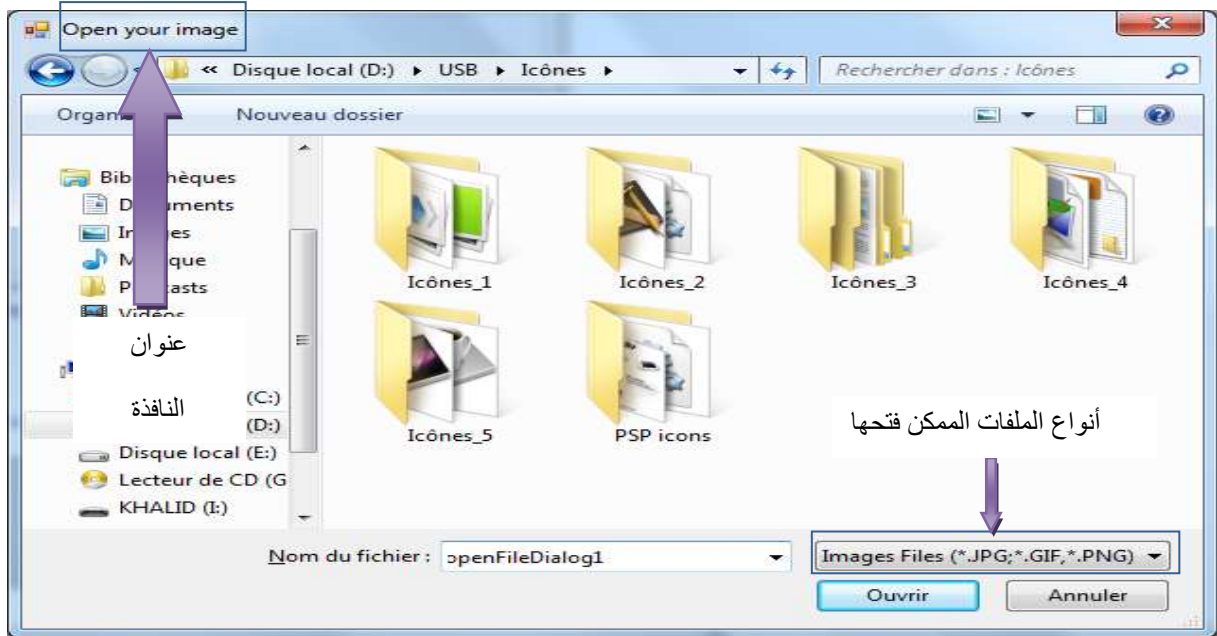
نفذ البرنامج واضغط على الزر لتطالعك نافذة فتح الملفات:

الآن تعرفت على شكل الأداة، الآن سنقوم بعرض خصائصها في الجدول التالي:

الخصيصة	دورها
FileName	تمكننا هذه الخصيصة من معرفة مسار الملف كاملا.
Filter	تستعمل هذه الخصيصة لفرز أنواع الملفات الممكن فتحها، مثلا فتح الملف على شكل صورة من نوع *.JPG أو *.GIF أو ملف نصي *.TXT ... ويكون شكل عملية فرز الأنواع كما يلي: Images Files|*.JPEG;*.PNG
InitialDirectory	لتحديد المسار الافتراضي الذي يظهر في بداية انطلاق نافذة الفتح SaveFileDialog
Title	عنوان نافذة الفتح.

قم بإضافة أداة OpenFileDialog وأداة PictureBox إلى الفورم، سنقوم بفتح صورة لتظهر على الأداة العارضة للصور:

الآن ادخل إلى الحدث Click الخاص بالزر واكتب فيه ما يلي:

```
openFileDialog1.Filter = "Images Files|*.JPG;*.GIF,*.PNG";
openFileDialog1.Title = "Open your image";
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
pictureBox1.Image = Image.FromFile(openFileDialog1.FileName);
}
```

في السطر الأول قمنا بتحديد أنواع الصور الممكن فتحها، قم بإضافة أنواع أخرى إذا أحببت ذلك.

في السطر الثاني أعطينا النافذة عنوانا.

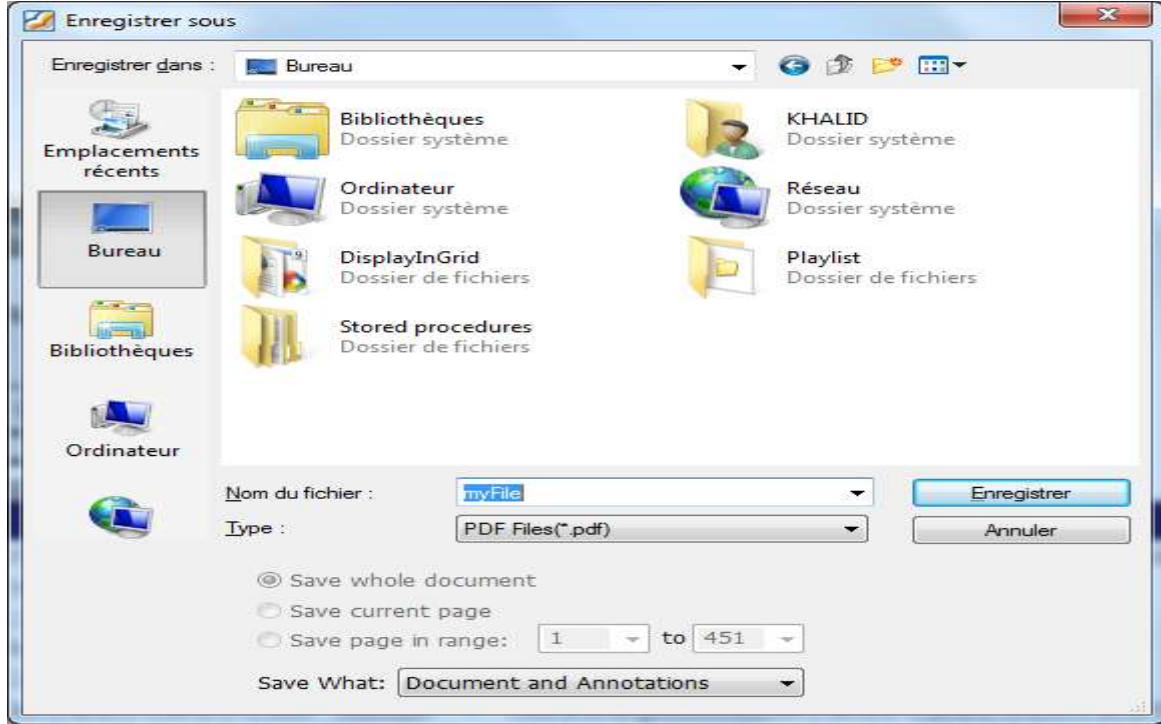
في السطر الثالث، نتحقق من أن المستخدم قد ضغط على زر OK، الذي معناه أنه قام باختيار الصورة، إذا تحقق هذا الشرط نظهر الصورة في أداة الصورة pictureBox1.

إذا قمت باختيار صورة فستكون النتيجة كهذه:

ماذا تلاحظ؟؟

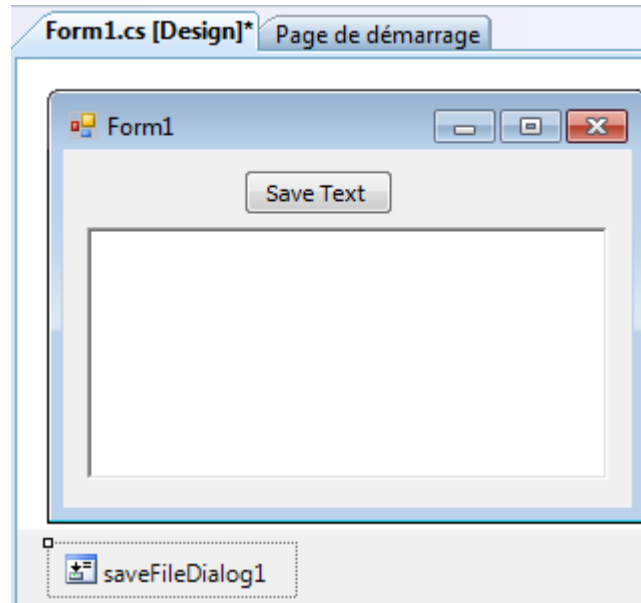
البقرة وإلا فلن يشتغل التطبيق...

34. أداة الحفظ SaveFileDialog

ستحتاج هذه الأداة إذا أردت القيام بعملية حفظ البيانات بمختلف أنواعها على حاسوبك، وهذه صورة للأداة عندما تستعملها في برنامجك:

هذا الجدول يعرض بعض الخصائص الأساسية لهذه الأداة:

الخصيصة	دورها
FileName	تمكن هذه الخصيصة من تحديد مسار الملف الذي تم حفظه عند الضغط على OK أي أن الإجراء ShowDialog أصبح يساوي DialogResult.OK
Filter	تستعمل هذه الخصيصة لفرز أنواع الملفات الممكن الحفظ بامتدادها، مثلا حفظ الملف على

الخاصية	الوصف
	شكل صورة من نوع JPG.* أو GIF.* أو ملف نصي TXT.* ... ويكون شكل عملية فرز الأنواع كما يلي: Images Files|*.JPEG;*.PNG
InitialDirectory	لتحديد المسار الافتراضي الذي يظهر في بداية انطلاق نافذة الحفظ SaveFileDialog
Title	عنوان نافذة الحفظ.

تقريبا نفس خصائص الأداة السابقة، قم بتصميم فورم كهذا:

بعد أن تنجز التصميم، ادخل إلى الحدث Click الخاص بالزر Save Text، واكتب فيه ما يلي:

```
            saveFileDialog1.Title = "Save Your Text";
            saveFileDialog1.Filter = "Text Documents | *.TXT";
            if (saveFileDialog1.ShowDialog() == DialogResult.OK)
            {
                System.IO.File.WriteAllText(saveFileDialog1.FileName,
richTextBox1.Text);
            }
```

طبعا في السطر الأول نقوم بتحديد عنوان النافذة.

في السطر الثاني نفرز الأنواع المقبولة، وفي حالتنا سيتم الحفظ على شكل ملفات نصية فقط.

في السطر الثالث، نتحقق من أن المستخدم قام بإدخال اسم الملف وتحديد مساره، إذا تحقق الشرط نقوم بحفظ محتوى الأداة richTextBox1 بواسطة مجال الأسماء الخاص بالملفات الذي رأيناه سابقا.

35. أداة متصفح المجلدات FolderBrowserDialog

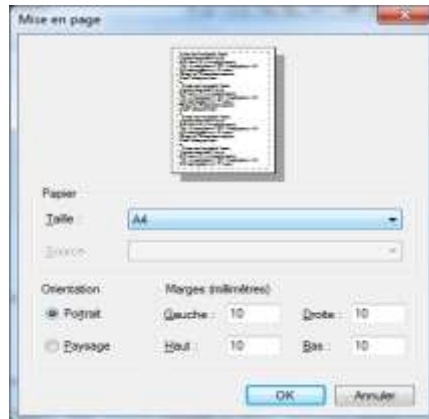
وتستعمل هذه الأداة لتحديد المجلدات Folders، وهذه صورة للأداة:

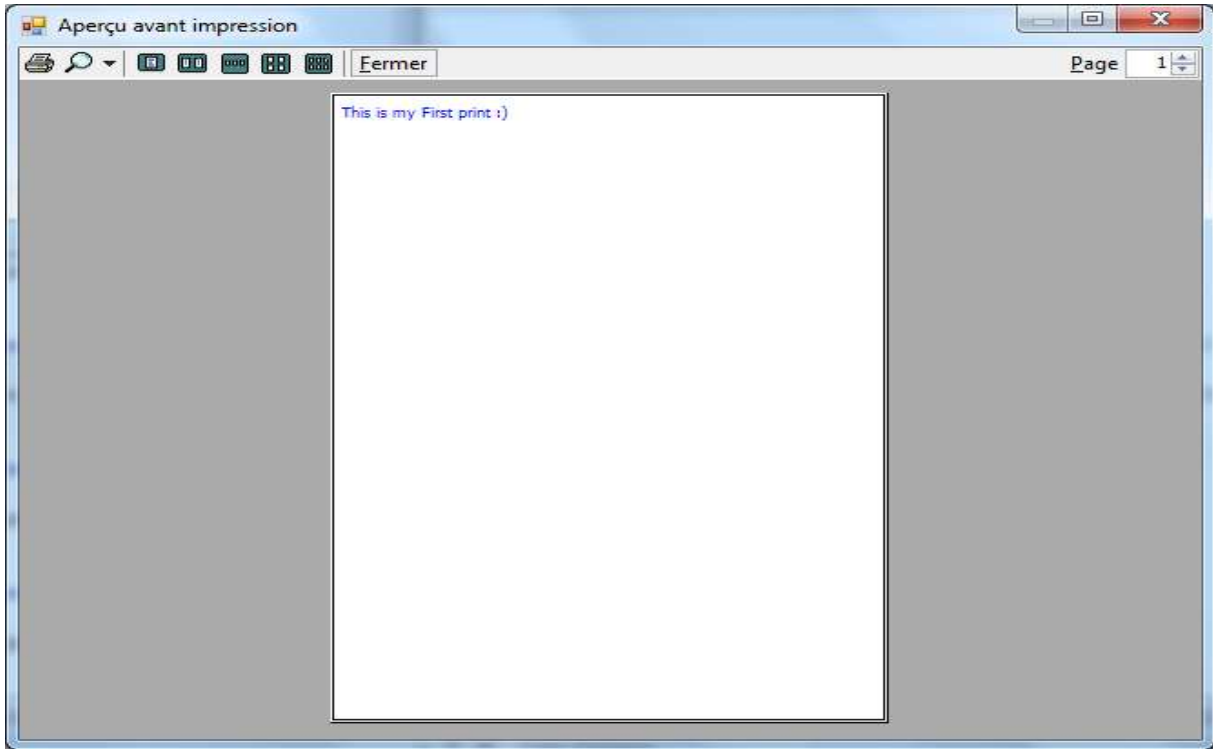
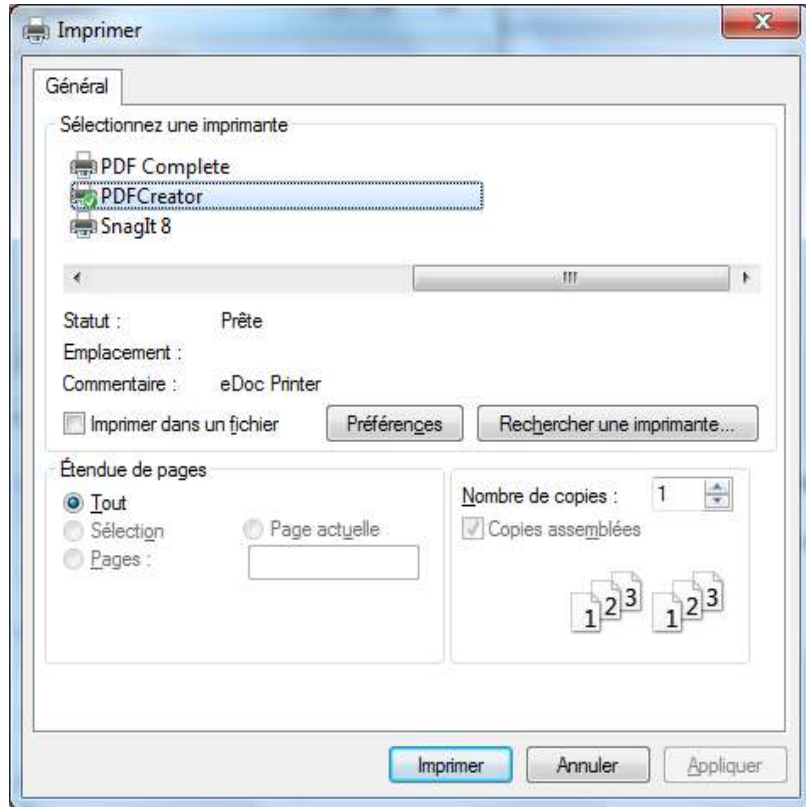
طريقة استخدامها شبيهة بما رأيناه مع الأدوات الأخيرة، سأضيف فقط أنه لمعرفة مسار المجلد المحدد نستعمل الخصيصة SelectedPath:

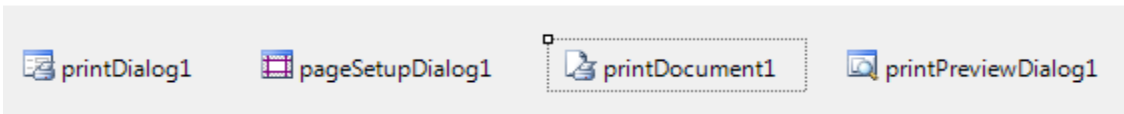
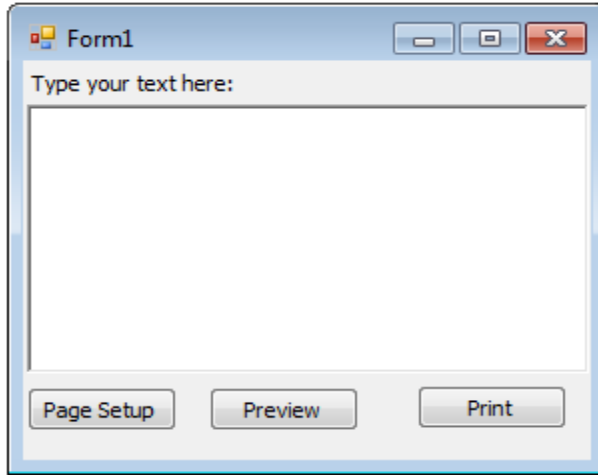
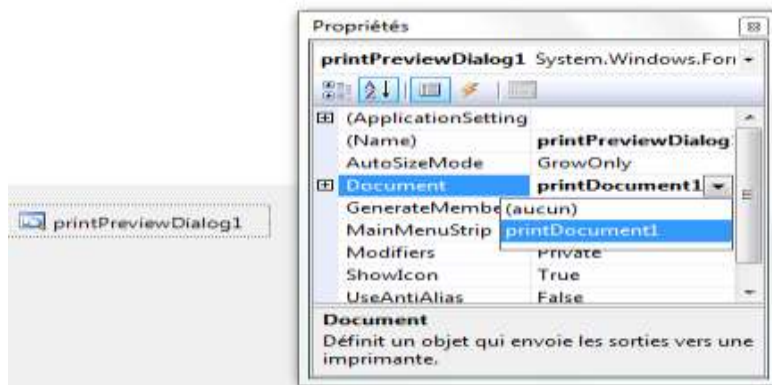
```
string folderPath = folderBrowserDialog1.SelectedPath;
```

36. أدوات الطباعة Print Tools

لاشك أن طباعة المحتوى جزء مهم من أي برنامج، بحيث يبقى ناقصا من دون هذا الأمر، توفر لنا لغة c# مجموعة من الوسائل لطباعة المحتوى، لعل أبرزها الأداة PrintDocument، وتقارير الكريستال Crystal Report؛ طبعا لن نتطرق إلى هذا النوع من التقارير لأن الحيز المناسب له هو الجزء الثالث من الكتاب الخاص بالتعامل مع قواعد البيانات Databases، سوف نقوم بطباعة المحتوى بواسطة الأداة PrintDocument وما يتعلق بها من أدوات وهي كما يلي:

PageSetupDialog: هذه الأداة تقوم بإظهار إعدادات الصفحة قيد الطبع، ولا شك أنك صادفت مثل هذه النافذة عند قيامك بعلية الطباعة:

من خلال هذه الأداة تستطيع تحديد حجم الصفحة وطريقة عرضها وما إلى ذلك من باقي الإعدادات.

 PrintPreviewDialog: وهي التي تقوم بتقديم صورة للصفحة المراد طباعتها:

PrintDialog: تقوم بإظهار نافذة الطباعة مباشرة لتمكين المستخدم من تحديد عدد الصفحات واختيار الآلة الطابعة في حال وجود أكثر من آلة طابعة، وهذه صورة للنافذة التي تظهرها الأداة:

PrintDocument: وهي أهم أداة في أدوات الطباعة، وكل الأدوات السابقة في حاجة إليها لأنها هي المحتوى المراد طباعته

قم بإنشاء مشروع جديد، وأضف إليه كل الأدوات السابقة، ستلاحظ بأن كل تلك الأدوات تظهر أسفل الفورم، ولا واحدة تظهر عليه، قم بإضافة الأداة RichTextBox لنكتب فيها النص المراد طباعته، وأضف أيضا ثلاثة أزرار: واحد لإعدادات الصفحة، والثاني لمشاهدة الصفحة قبل الطبع، والثالث للطبع مباشرة، كما تعرض الصورة التالية:

جيد، الآن سنقوم بالربط بين كل الأدوات وبين الأداة printDocument1، وكأننا نقول لهذه الأدوات هذا هو المحتوى الذي عليكم طباعته، طريقة الربط سهلة جدا، نحدد كل أداة من أدوات الطباعة، ونذهب إلى نافذة الخصائص، وبالضبط إلى الخصيصة Document، وفيها نقوم بتحديد الأداة printDocument1:

الآن أنهينا التصميم، بقي لنا فقط كتابة بعض الأسطر من الكود لكي يعمل البرنامج على أحسن وجه، قبل ذلك ألفت انتباهكم إلى أن الأداة PrintDocument تتوفر على الحدث PrintPage الذي يتولد عند كل عملية طباعة، بمعنى لو أننا أردنا كتابة كود الطباعة فعلينا كتابته في هذا الحدث.

قم بالدخول إلى هذا الحدث، عن طريق تحديد الأداة PrintDocument ثم الانتقال إلى نافذة الخصائص ومنها إلى نافذة الأحداث.

اكتب السطر التالي:

```
e.Graphics.DrawString(richTextBox1.Text, new Font("Tahoma", 18), Brushes.Blue,10,20);
```

الدالة DrawString التابعة للفئة Graphics، تقوم برسم نص، وستجد العديد من الدوال منها ما يقوم برسم مستطيل ومنها ما يقوم برسم قوس وغيرها كثير، تستقبل هذه الدالة أربع برامترات وهي كالتالي:

- النص المراد طباعته: في حالتنا هذه سيكون هو النص المكتوب في الأداة RichTextBox
- الخط: وفي هالتنا هذه حددنا الخط Tahoma والحجم 18
- لون الفرشاة: قمنا باختيار اللون الأزرق.
- إحداثيات بداية كتاب النص: وهي أقصى اليسار في الأعلى وبإمكانك اختيار أية إحداثيات لظهور النص في الصفحة.

الآن قمنا بإعطاء الأداة PrintDocument محتوى الطباعة ولونه وخطه وتمركزه.

قم بالولوج إلى الحدث Click الخاص بالزر الأول Page Setup واكتب فيه هذا السطر:

```
pageSetupDialog1.ShowDialog();
```

رأينا مثله مرار وتكرارا أليس كذلك؟؟ إنه بكل بساطة يقوم بإظهار نافذة إعدادات الصفحة عند الضغط على هذا الزر.

ادخل إلى الحدث Click الخاص بالزر الثاني Preview واكتب فيه مايلي:

```
printPreviewDialog1.ShowDialog();
```

وفي الحدث Click للزر الثالث، اكتب ما يلي:

```
printDialog1.ShowDialog();
```

الآن قم بتنفيذ البرنامج وشاهد النتيجة.

طيب هذا فيما يخص طباعة النص، ماذا عن الصور ؟؟؟

لا عليك الأمر بسيط جدا، الدالة DrawString تقوم بطباعة النص، إذن فلا شك أن الفئة Graphics تحتوي على دالة لطباعة الصور أيضا.. تعال معي !

إذا بحثت بين دوال الفئة Graphics ستجد هذه الدالة DrawImage، وهي دالة خاصة برسم الصور وتستقبل برامترين، الأول لتحديد مكان الصورة المراد إظهارها، والثاني لتحديد إحداثيات تموضع الصورة، وفي هذا المثال نقوم بطباعة الصورة:

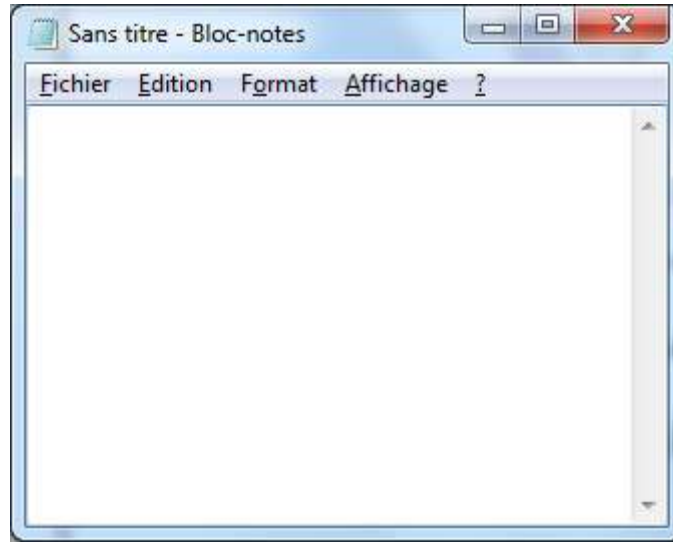
```
e.Graphics.DrawImage(Image.FromFile("c:\\c.png"), new Point(10, 10));
```

قم بكتابة هذا السطر مكان السطر الخاص بإظهار النصوص، وستلاحظ أن الصورة التي كتبنا مسارها سيتم طباعتها، هذه أمثلة سطحية الغاية منها الاستئناس بهذه الأدوات، لأنك حينما تفهم هذه المسائل ستكون قادرا حتما على الإبداع فيها كما تشاء.

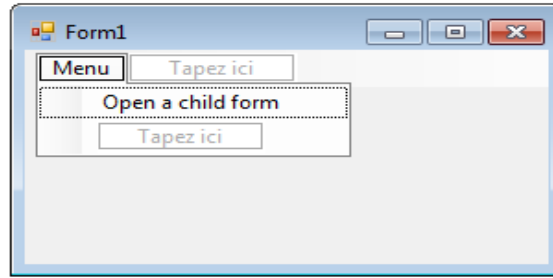
37. التطبيقات متعددة النوافذ MDI وأحادية النوافذ SDI

تحديدا يوجد ثلاثة أنواع من التطبيقات الممكن إنشاؤها في برمجة الواجهة:

- Dialog-Based Application (DBA): ويكون البرنامج عبارة عن نافذة واحدة مثل برنامج عارض الصور المرفق مع الويندوز:

- Single-Document Interfaces (SDI): أيضا يكون هذا البرنامج يضم نافذة واحدة ولكنه يتوفر على قوائم Menu، وكمثال على ذلك برنامج المفكرة Notepad:

- Multiple-Document Interfaces (MDI): ويكون التطبيق من هذا النوع يضم العديد من النوافذ التي يمكنك إظهارها في أي وقت تشاء، وكمثال على ذلك برنامج الفوتوشوب أو برنامج الفيجوال استوديو.

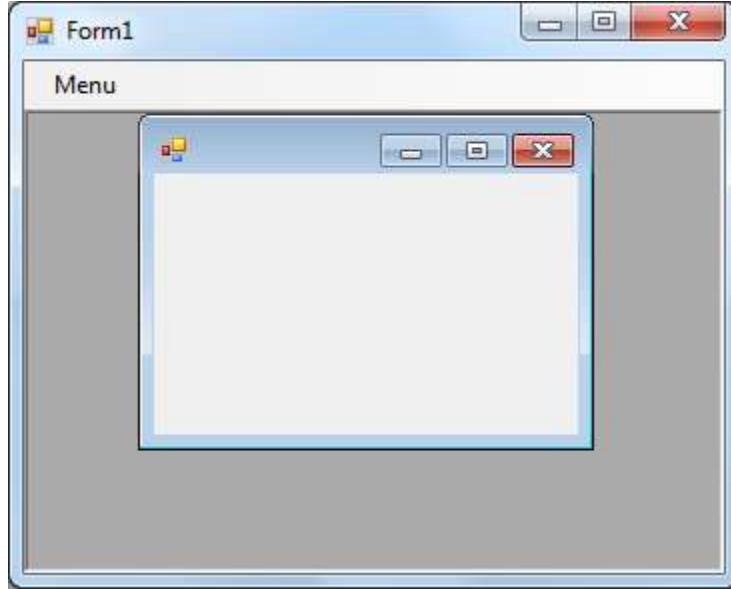
بالنسبة للنوعين الأولين فليس ثمة داعٍ لإنشاء نماذج لها، لأننا تعاملنا معها في كل الأمثلة التي عرضناها، أما النوع الثالث وهو ما يهمنا فسوف ننجز تطبيقا بسيطا للتعرف عليه أكثر.

قم بإنشاء مشروع جديد، وأضف إليه أداة القائمة MenuStrip، كما تظهر الصورة التالية:

قم بتحديد الفورم، ثم اذهب إلى نافذة خصائصه، وابحث عن الخصيصة IsMdiContainer واجعل قيمتها تساوي True، ستلاحظ أن لون الفورم صار داكنا، وذلك دلالة على أنه صار هو الفورم الحاضن MDI Container أو الأب MDI Parent.

جيد، إلى هنا الأمور كُوَيِّسَة كما يقول إخوتنا المصريون

اذهب إلى القائمة الفرعية Open a child form واضغط عليها مرتين لتنتقل إلى الحدث Click الخاص بها، وهناك اكتب ما يلي:

```
Form frmChild = new Form();
frmChild.MdiParent = this;
frmChild.Show();
```

قمنا في السطر الأول بإنشاء فورم جديد اسمه frmChild، ثم بعد ذلك قمنا بتحديد الفورم الأب لهذا الفورم والكلمة المحجوزة this تعني الفورم الحالي كما مر معنا، وفي السطر الثالث نقوم بإظهار هذا الفورم الابن، وستكون النتيجة هكذا:

لاحظ عند نقلك للفورم الابن من مكان إلى مكان أنه لا يستطيع تجاوز حدود الفورم الأب...ونعم التربية 😊

لكن ماذا لو كان لدينا الفورم جاهز مسبقا ونريد أن نجعله ابنا لفورم آخر؟

الأمر سهل جدا، لنفترض أنه لدينا فورم1 و فورم2، نريد أن نجعل الأول أبا والثاني ابنا، اتبع الخطوات التالية:

- اجعل الفورم الأول حاضنا بواسطة الخصيصة IsMdiContainer، غير قيمتها نحو True
- قم بالإعلان عن متغير من نوع الفورم الثاني في الحدث الذي تريد إظهار هذا الفورم فيه، ثم قم بتحديد الفورم الحاضن له بواسطة الخاصية MdiParent، كما سأورد هنا:

```
Form2 frmChild = new Form2();
frmChild.MdiParent = this;
frmChild.Show();
```

ستصل إلى نفس النتيجة الأولى إن شاء الله.

الخـــاتـــمـــة

الحمد لله ابتداء وانتهاء، والصلاة والسلام على حبيبنا محمد صلى الله عليه وعلى آله وصحبه وسلم تسليما كثيرا، وبعد:

أزف إلى إخواني وأخواتي بشرى أن الجزء الثالث من كتاب سبيلك المختصر سيتحدث عن قواعد البيانات وسيكون عاما وشاملا إن شاء الله.

كما يسرني أن أخبركم أنني بصدد الإعداد لموقع تعليمي يشمل معظم لغات البرمجة، وسيكون الموقع بثلاث لغات وهي (العربية، الفرنسية والإنجليزية)، بالنسبة للدروس سأشرف عليها شخصيا وستكون بصيغة مقروءة وبصيغة مرئية على شكل فيديوهات، أعلم أن الأمر شاق ومتعب ولكن نسأل الله المدد والعون، وسأضع في الموقع إن شاء الله جزء خاصا بمشاريع التخرج لمساعدة من هم في حاجة إلى مساعدة وكذلك سيكون تحت كل درس فضاء لطرح الأسئلة وللمشاركة باقتراحاتكم، إن أعجبتكم فكرة الموقع راسلوني عبر البريد الالكتروني واذكروا لي آراءكم و توجيهاتكم.

Khalid_ESSAADANI@Hotmail.Fr